MULTIPLE MODEL ADAPTIVE CONTROL OF THE VISTA F-16

THESIS

Michael Joseph Stepaniak
Second Lieutenant, USAF

AFIT/GE/ENG/95D-26

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/95D-26

MULTIPLE MODEL ADAPTIVE CONTROL OF THE VISTA F-16

THESIS

Michael Joseph Stepaniak
Second Lieutenant, USAF

AFIT/GE/ENG/95D-26

AFIT/GE/ENG/95D-26

MULTIPLE MODEL ADAPTIVE CONTROL OF THE VISTA F-16

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Michael Joseph Stepaniak, B.S.E.E.

Second Lieutenant, USAF

December, 1995

Acknowledgements

First and foremost I would like to thank my wife, Annie, for being there through it all. Her patience and support made the long hours tolerable. I was (and am) blessed to have her at my side. At the same time, I could not have asked for a better thesis advisor than Dr. Peter Maybeck. It was a very rewarding experience to have worked with him, and his guidance made this thesis possible. I would be remiss if I did not mention my fellow guidance and control classmates. Without their comic relief, life at AFIT would have been a most unpleasureable experience. Thanks also to my family who has always provided the encouragement for me to keep going. Last, but certainly not least, I thank God for staying up with me late into the night, for keeping me focused, and for seeing me safely through to the end.

Michael Joseph Stepaniak

Table of Contents

Page

List of Figures

Figure Page

List of Tables

AFIT/GE/ENG/95D-26

Abstract

Multiple model adaptive control (MMAC) is investigated using the high-fidelity, nonlinear, six-degree-of-freedom Simulation Rapid-Prototyping Facility VISTA F-16. Detection of single actuator and sensor failures is considered, with an MMAC algorithm initially pursued which allows a controller specifically designed for each particular failure condition to replace the standard F-16 Block 40 flight control system (FCS) once the failure is detected. The synthesis of certain discrete-time LQG/PI controllers (those using control variables linearly dependent on state derivatives) is shown to be unattainable due to numerical difficulties. A novel control technique, termed control redistribution, is introduced which redistributes control commands (that would normally be sent to failed actuators) to the non-failed actuators, accomplishing the same control action on the aircraft. Multiple model adaptive estimation-based control redistribution is demonstrated to detect single failures in less than one second and to provide a response nearly identical to that anticipated from a fully functional aircraft in the same environment. Moreover, this method directly employs the proven Block 40 FCS, and no other, thereby guaranteeing desirable closed loop performance. A description of modifications necessary for in-flight testing is also provided. This research represents the most realistic simulation of multiple model adaptive control for flight control to date.

MULTIPLE MODEL ADAPTIVE CONTROL OF THE VISTA F-16

1. Introduction

1.1 Chapter Overview

This thesis details the implementation of a Multiple Model Adaptive Controller for the VISTA F-16 confronted with failed actuators and sensors. This chapter will first provide the motivation for an MMAC. Section 1.3 will then present the problem statement and define the scope of the research. Major assumptions of this research will be noted in Section 1.4. Finally, a reference format for the rest of this document is provided, and a chapter summary given.

1.2 Motivation

In designing flight control systems, aircraft are often modeled as deterministic, linear time invariant systems [3,38]. In reality, many uncertain parameters exist in these models, as well as all other real world models. For example, sensor measurements are always noise corrupted, and system performance often changes as the true operating condition varies over time. Even if the model were somehow to account for these uncertainties, physical components generally degrade over time, and sometimes fail. Furthermore the model itself is always suspect; the real world is never truly linear, and the higher order terms neglected during linearization can be cast as uncertain parameters. At issue then is the ability of the controller to compensate for changes in system parameters and maintain stable flight with as much a preservation of handling qualities as possible.

Multiple model adaptive control (MMAC) is a modern technique that allows for continuous, real-time adaptation to parameter variation. Of particular interest in flight control is the ability to compensate for changes due to damage, degradation, or complete failure of the sensors and/or actuators. In doing so, the MMAC may lessen the reliance on redundant hardware systems, freeing up the limited payload for more critical functions. MMAC also enhances mission effectiveness by allowing crippled aircraft to maintain handling qualities so they may either continue the mission, return to base for repairs, or, as a minimum, provide a stable platform for safe egress of the air crew. While other robust techniques strive to meet the same goals, MMAC potentially provides greater levels of performance for a wide range of flight conditions.[1]

1.3 Problem Statement

Multiple Model Adaptive Control provides the ability to detect changes in the uncertain parameters, and reconfigure in real-time the control law to compensate for these changes. Previous research has already demonstrated that multiple model techniques are capable to detecting and isolating actuator and sensor failures [31,35]. Further, MMAC's have been shown to provide stable, satisfactory flight in the face of single and most double failures[2] when tested against linear truth models [20,31]. An MMAE-based controller [34] was also able to maintain a stable, straight and level flight trajectory despite failures when verified against a nonlinear truth model. This research will demonstrate that a multiple model structure is capable of providing appropriate responses to pilot commands in the face of sensor and actuator failures by implementing two forms of multiple model adaptive controllers for the VISTA F-16 and verifying their performance on a nonlinear, six-degree-of-freedom simulation. The first implementation is an MMAC using LQG/PI controllers designed to match the performance of the Block 40 flight control system (FCS). The second implementation is an MMAE-based controller using a novel control technique, termed control redistribution, which is first presented in Section 3.8 of this thesis. These demonstrations will provide critical data ascertaining the MMAC's ability to detect and compensate for actuator and sensor failures in as realistic a scenario as possible.

[1]Gain scheduling is used to provide control throughout the flight envelope. In the event that the scheduled variable, typically dynamic pressure, is also considered an uncertain parameter, a multiple model adaptive estimator could be used to provide estimates of the required measurements to be used as inputs to the flight control system [2].

[2]There are certain cases where the loss of two control surfaces does not leave the airframe with enough control authority to provide stable flight under any circumstances.

1.4 Assumptions

Although the design is intended for the VISTA F-16, the truth model used in this research is the SRF VISTA F-16 simulation [15,16]. The verification of this simulation as a valid model of the true aircraft is beyond the scope of this thesis. The accuracy of this model is therefore assumed acceptable, at least for the duration of the eight second runs used for performance analysis. The failure models generated intuitively appear to be reasonable models of true failure conditions. Results derived from these failure models will presumably be indicative of the performance of the actual VISTA F-16 experiencing real failures. Other assumptions will be addressed throughout the following chapters as they arise.

1.5 Thesis Format

Chapter 1 has provided a brief motivation into the need for a multiple model adaptive controller. Chapter 2 will begin with a chronological summary of the contributions to the multiple model design, followed by an analysis of the elements of an MMAC/MMAE. The concept of control redistribution is also introduced. A list of all modifications made for implementation is then provided. The truth and design models are presented in Chapter 3, along with derivations of the LQG controller and control redistribution. The results of the research will be given in Chapter 4, followed by conclusions and recommendations in Chapter 5. The appendices will provide additional information so that the research and its results may be easily duplicated.

1.6 Chapter Summary

This chapter serves as a motivation for adaptive control of sensor and actuator failures, and gives an introduction to the point of research: synthesis of a multiple model adaptive controller. The problem statement, scope, and major assumptions of this research were then defined, and a brief format of the entire document was provided for reference. Chapter 2 will begin to flesh out the concepts introduced in this chapter.

2. Algorithm Development

2.1 Chapter Overview

This chapter provides a brief synopsis of the developments leading up to the current research on multiple model adaptive control (MMAC). An overview is then provided in Section 2.3 to acquaint the reader with the basic MMAC structure, and to introduce the two types of controllers which will be implemented in this research. Section 2.4 covers issues related to design implementation, with modifications detailed as necessary. This chapter concludes with a chapter summary in Section 2.5.

2.2 Summary of Current Knowledge

2.2.1 History of Multiple Model Techniques. The concept of a multiple model structure was first introduced by Magill in 1965 [19]. Though not referred to as a multiple model adaptive estimator (MMAE), his design was strikingly similar, and his notion that an "optimal adaptive estimate is an appropriately weighted summation of conditional estimates which are formed by a set of elemental estimators" serves as the foundation to the MMAE design. Later Lainiotis incorporated elemental controllers [17] to develop the first multiple model adaptive controller. At this time multiple model research remained purely theoretical [2], though Lainiotis realized that an implementation would be ideally suited for parallel processors. Also, only constant parameters could be considered since the recursive probability calculations allowed the computed conditional probabilities of assumed discrete values for parameters being correct to go to zero, i. e. the system was learning the uncertain parameters so well that it became insensitive to change.

In 1977 a group of researchers from the Massachusetts Institute of Technology, led by Athans, developed the first MMAC implementation [2], coincidentally in a flight control application, though they estimated flight condition rather than failure mode. However, the importance of their findings cannot be overlooked. First, they introduced a lower bound on the probability calculations, increasing response times and allowing the system to respond to time-varying parameters. Second, the presence of "beta dominance", discussed in Section 2.4.2, was first made known. Also a weighted average (or Bayesian method) was used to calculate the control signal. Finally, the need for a test input, or dither, to excite the system to promote identification was addressed. Equally important was the realization that this dither should be subliminal so as not to impact the pilot and his mission adversely. Unfortunately, a valid assessment of the MMAC's effectiveness could not be made due to limitations in the experiment. For example, the NASA-supplied F-8 which served as the testbed, was outright inherently stable and easy to control, and therefore not a particularly challenging problem for any type of adaptive control. Also, the computer technology for digital flight control systems was not yet advanced enough to allow a full implementation. As a result, only four models were implemented (out of the sixteen which were designed), each operating at a slow sampling rate of 8 Hz (compared to 64 Hz for VISTA F-16).

Chang and Athans [6] later revealed that the Bayesian method of estimate or control calculation was only optimal provided the true parameter value matched one of the assumed hypotheses and provided the unknown parameter remained constant. The earlier implementation on the F-8 was therefore suboptimal. Although Chang and Athans attempted to extend optimality to include Markov 1 processes (time-varying processes where knowledge of the current time step is sufficient to completely characterize the transition to the next step), their algorithm was shown by Tugnait [49] to be suboptimal again. However, because the number of hypotheses required for the optimal design grows exponentially with the number of possible parameter realizations [6], only suboptimal designs are realizable for most problems anyway.

The computational resources required for implementation has always been prohibitive, though the use of parallel processor greatly reduces the apparent load [10, 17]. Fry and Sage [10] investigated the use of a hierarchical structure to reduce computational loading further. Used for system identification, they broke the problem into separate subsystems, with one unknown parameter per subsystem, all coordinated by a "supremal" unit. By reducing the number of on-line filters needed at any one time, the multiple model structures became more plausible for implementation on a real world system with many uncertain parameters. In the context of flight control, sensor and actuator failures are well suited to hierarchical structures in which the MMAC swaps in and out different banks of filters/controllers. For example, the first bank may contain Kalman filters and associated controllers tuned to look for single failures. Should a failure occur, perhaps in the rudder, the MMAC swaps out the entire filter/controller bank in exchange for a second bank containing Kalman filters and associated controllers tuned for two failures, one of which is known to be the rudder. This second bank also contains a filter embedded with the fully functional hypothesis so that the MMAC can reverse a declaration (in the event of a false alarm or intermittent failure) and return back to the first bank. With eleven possible sensor and actuator failures, a single bank without the hierarchical structure would require 67 filter/controller pairs to account for all possible single and/or double failures. Hierarchical structuring, on the other hand, reduces the number of filter/controller pairs to only twelve at any one time, all of which can be running on separate, parallel processors so that the effective computational loading is that of only one Kalman filter, one controller, and the probability evaluator which contains the detection and switching logic.

2.2.2 Contributions at AFIT. The Air Force Institute of Technology (AFIT) is a driving force behind the research into MMAC. A 1985 contribution by Maybeck and Suizu [22] alleviated the beta dominance effect in a very different problem context by removing the β term altogether. This solution is problem dependent though, because some systems may rely in part on the β term to distinguish between models. In Section 2.3.1, the β term will be shown to be inversely proportional, for a linear time-invariant system, to the determinant of $\mathbf{H}\mathbf{P}(t_i^-)\mathbf{H}^T + \mathbf{R}$, and Section 2.4.2 will fully discuss its importance for sensor failure detection. Later, Stevens [31] and Menke [35] investigated scalar residual monitoring as an alternate solution to the beta dominance problem.

In 1987, Maybeck and Hentz [27] considered reducing the total number of online filters (or filter/controller pairs) in a multiple model algorithm by means of a moving bank of filters; this is analogous to the hierarchical structure pursued specifically in flight control applications later [8, 31, 35]. Of particular concern to them was the logic needed to govern the movement of the bank. Other researchers used moving banks of elemental filter/controllers in an MMAC structure to quell vibrations in flexible space structures [1, 30]. Another application of multiple model algorithms at AFIT is target tracking using infrared measurements and laser illumination [28].

The first implementation of MMAC for flight control applications at AFIT was by Pogoda [29] in 1988. Pogoda's analysis focused on detecting sensor and actuator failures on the longitudinal channel of the STOL F-15. A year later, Stevens [31, 46] furthered the STOL F-15 research by investigating partial and multiple failures. Partial failures are controlled through the use of Bayesian blending [6, 24], discussed in Section 2.3.2, which uses a probability weighted average to blend the control laws from two or more elemental controllers, thereby accounting for failure conditions not explicitly hypothesized by any one filter/controller pair in the MMAC. More importantly, Stevens verified that removing the troublesome beta term, discovered by Athans, did indeed eliminate the false alarms caused by beta dominance. In fact, the modified MMAC responded faster to failures in general, and the probabilities displayed smaller steady-state standard deviations, than in the case of the unmodified MMAC. Stevens' results also revealed that more work was needed to detect partial failures, but that, even before removing the beta term, single and double failures yielded good detection and a stabilized aircraft with satisfactory control in approximately 93% of the test cases. For the remaining cases, the majority could be countered through additional voting or removing the beta term, and the few remaining exceptions involved double failures in which the aircraft was rendered with inadequate control authority to remain stable with any controller.

Following the STOL F-15 research, Martin [20] successfully implemented elemental controllers for an MMAC for both the longitudinal and lateral/directional channels of the AFTI F-16. Controllers for nine points in the flight envelope were designed, with special emphasis placed on high angle-of-attack. The elemental controllers were synthesized via the LQG design method, incorporating both implicit and explicit model following. Both the LQG synthesis and model following will be described in more detail in Chapter 3. Again, and for all continuing research at AFIT, the uncertain parameter is the failure status of the aircraft: varying from a fully functional aircraft to allowing sensor and actuator failures.

One problem plaguing the MMAC and MMAE-based controllers is that of observability. For instance, an aircraft flying straight and level is not using the rudder. As a result, the MMAC or MMAE-based controller will not detect a damaged rudder until the pilot initiates maneuvers. If defensive maneuvers requiring severe turns are commanded, the control system may not have enough time to detect and compensate for the failed rudder to give the pilot the proper response that he commands. A solution is to apply a dither signal which moves all control surfaces, exciting the aircraft states enough to facilitate detection. Stratton and Menke [35,48] used an MMAE-based controller on the VISTA F-16 to investigate the effective use of dither to enhance failure detection. With a sinusoidal dither signal, Menke was able to detect all single failures, most double failures, and many partial failures. Stratton and Menke also investigated scalar residual monitoring (breaking the residual vectors from the Kalman filters into their component scalars) to provide corroborating votes for a failure declaration.

Next, a series of implementations [18, 26] on the LAMBDA Unmanned Research Vehicle (URV) investigated probability smoothing to reduce occurrences of false alarms, increased residual propagation to enhance detection by allowing the residuals to grow in size before updating the Kalman filters, and increasing the scalar penalty for measurement residuals to increase the speed of response. Note that, unlike the STOL F-15 and AFTI F-16 research, the VISTA F-16 and LAMBDA URV controllers did not use an MMAC, but rather they relied on MMAE-based control. The difference between these two control methodologies will be further discussed in Section 2.3.3.

Most recently, Eide [8] successfully implemented an MMAE-based controller on the six degree-of-freedom Simulation Rapid-Prototyping Facility (SRF) simulation for the VISTA F-16. The SRF VISTA simulation includes complete nonlinear lateral/directional and longitudinal dynamics. The F-16's Block 40 flight control system, including the aileron-rudder interconnect, is encoded as the controller. All actuator saturations and rate limits are also included in the simulation. Eide was able to demonstrate convergence to the correct failure hypothesis for exhaustive cases of both single and double actuator and sensor failures. This SRF implementation represents the most thorough and realistic simulation and analysis of multiple model flight control techniques to date, and is the basis for this research effort.

2.3 MMAC Overview

If all parameters of a system were perfectly known, then an ideal model could be created and there would be little motivation to pursue a multiple model structure. Unfortunately, in the real world, uncertainties exist even in the best of models. The multiple model synthesis provides the means to estimate the uncertain parameters accurately and then adapt the control signal accordingly.

The foundation of multiple model techniques is the discretization of the generally continuous and infinite space of uncertain parameters, such as a p-dimensional Euclidean space with the uncertain parameter vector denoted $\mathbf{a}$, into a finite number of representative points, denoted $\mathbf{a}_1$, $\mathbf{a}_2, \ldots, \mathbf{a}_K$ [25]. Each of these points represents an implicit hypothesis concerning the realization of the uncertain parameters. A bank of estimators is then designed, with each discrete point tied to an estimator embedded with the corresponding hypothesis. Based on the residual difference between measured sensor values and the predicted measurements output by these estimators, a determination is made on the probability of each hypothesis being true, and a state estimate is formed as a probability-weighted average of the individual filter state estimates. The resulting structure, pictured in Figure 2.1, is referred to as a multiple model adaptive estimator (MMAE). If the output, a state estimate, is used to drive a controller, the result is then termed an MMAE-based controller. Alternatively, a multiple model adaptive controller (MMAC) pairs each estimator, and its associated hypothesis about the parameter realization, with a separate controller. The output of an MMAC is then a probability-weighted control to be used as the commanded input into the dynamic system. Diagrams and a comparison of the MMAC and MMAE-based controllers will be presented later in Section 2.3.3.

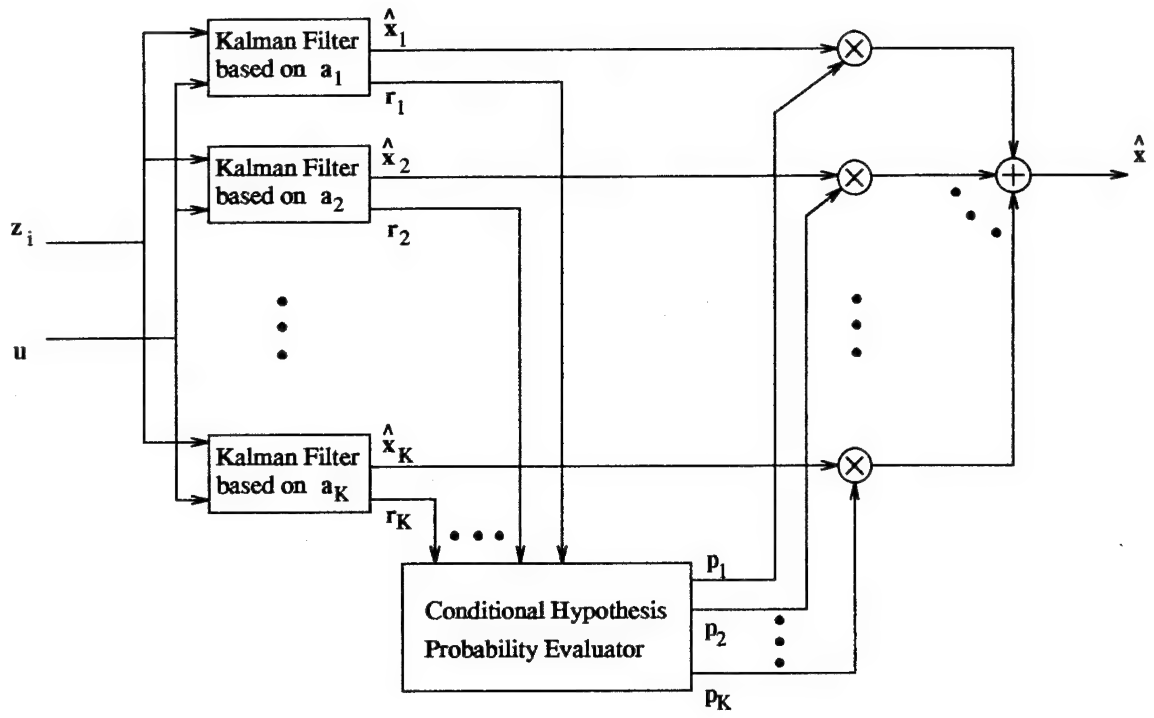

Kalman Filter based on a_1
$\hat{x}_1$
r_1
Kalman Filter based on a_2
$\hat{x}_2$
r_2
Kalman Filter based on a_K
$\hat{x}_K$
r_K
z_i
u
$\hat{x}$
Conditional Hypothesis Probability Evaluator
p_1
p_2
p_K

Figure 2.1 Multiple Model Adaptive Estimator

For the sequence of research on the VISTA F-16, the uncertain parameters in question span the space of all possible failure modes. The discretized space contains twelve points: a fully functional hypothesis, a failure hypothesis for each of five control surfaces, and a failure hypothesis for each of six sensors. The control surfaces are the left and right stabilators, the left and right flaperons, and the rudder. The leading edge flap is omitted primarily due to its lack of control authority at the design point chosen for this research. The sensors are chosen based on the requirements of the VISTA F-16's Block 40 flight control system. The measured variables are angle of attack, pitch rate, normal acceleration, roll rate, yaw rate, and lateral acceleration.

The estimators and controllers have been selected as Kalman filters and LQ (linear system, quadratic cost) optimal controllers. The Kalman filters are already designed for the MMAE-based controller previously implemented on the SRF simulation of the VISTA F-16. The LQ synthesis provides deterministic, optimal, full-state feedback controllers. An important aspect of LQ controllers is that they can easily be designed independent of the filters due to the separation principle [24] of LQG (linear, quadratic, Gaussian) optimal stochastic control, discussed later in Section 3.6. An alternate method of control, MMAE-based control redistribution, will also be introduced. Finally, a Bayesian approach is used for the probability evaluator. These three parts of the MMAC, the elemental Kalman filters, the elemental LQ controllers, and the conditional hypothesis probability evaluator, as well as MMAE-based control redistribution, will be discussed in more detail in the remainder of this chapter.

2.3.1 Kalman Filters. The estimators used in this research are Kalman filters. These filters use the control signals as inputs to propagate an internal model, which is based on the corresponding assumed failure hypothesis. The Kalman filter then performs an update cycle, merging the internally computed states and the information from system measurements to form a state estimate, $\hat{\mathbf{x}}(t_i^+)$, and a residual vector, $\mathbf{r}(t_i)$, defined in Equation (2.7) as the difference between the measured value and the filter predicted value of that measurement. A complete derivation can be found in *Stochastic Models, Estimation, and Control Volume 1* [23]; only the resulting equations will be summarized here.

The dynamics model is assumed to be a linear time-invariant, discrete-time system of the form:

$$\begin{aligned} \mathbf{x}(t_i) &= \mathbf{\Phi}\mathbf{x}(t_i) + \mathbf{B}_d\mathbf{u}(t_i) + \mathbf{w}_d(t_i) \\ \mathbf{z}(t_i) &= \mathbf{H}\mathbf{x}(t_i) + \mathbf{D}_z\mathbf{u}(t_i) + \mathbf{v}(t_i) \end{aligned} \tag{2.1}$$

where $\mathbf{x}$ is the vector of system states, $\mathbf{u}$ the control input vector, $\mathbf{z}$ the measurement vector, and where dynamics driving noise, $\mathbf{w}_d(t_i)$, and measurement corruption noise, $\mathbf{v}(t_i)$, are discrete-time, white Gaussian noises with the statistics:

$$\begin{array}{ccc} E\left[\mathbf{w}_d(t_i)\right]=\mathbf{0} & E\left[\mathbf{w}_d(t_i)\mathbf{w}_d^T(t_i)\right]=\mathbf{Q}_d & \\ & & E\left[\mathbf{w}_d(t_i)\mathbf{v}^T(t_i)\right]=\mathbf{0} \\ E\left[\mathbf{v}(t_i)\right]=\mathbf{0} & E\left[\mathbf{v}(t_i)\mathbf{v}^T(t_i)\right]=\mathbf{R} & \end{array} \tag{2.2}$$

The k^{th} elemental filter in the multiple model structure is propagated forward by

$$\hat{\mathbf{x}}_k(t_{i+1}^-) = \mathbf{\Phi}_k\hat{\mathbf{x}}_k(t_i^+) + \mathbf{B}_{dk}\mathbf{u}(t_i) \tag{2.3}$$

$$\mathbf{P}_k(t_{i+1}^-) = \mathbf{\Phi}_k\mathbf{P}_k(t_i^+)\mathbf{\Phi}_k^T + \mathbf{G}_{dk}\mathbf{Q}_{dk}\mathbf{G}_{dk}^T \tag{2.4}$$

starting from the initial conditions, $\hat{\mathbf{x}}(t_0)$ and $\mathbf{P}(t_0)$. The filter is updated by

$$\mathbf{A}_k(t_i) = \mathbf{H}_k\mathbf{P}_k(t_i^-)\mathbf{H}_k^T + \mathbf{R}_k \tag{2.5}$$

$$\mathbf{K}_k(t_i) = \mathbf{P}_k(t_i^-)\mathbf{H}_k^T\mathbf{A}_k^{-1}(t_i) \tag{2.6}$$

$$\mathbf{r}_k(t_i) = \mathbf{z}_i - \mathbf{H}_k\hat{\mathbf{x}}_k(t_i^-) \tag{2.7}$$

$$\hat{\mathbf{x}}_k(t_i^+) = \hat{\mathbf{x}}_k(t_i^-) + \mathbf{K}_k(t_i)\mathbf{r}_k(t_i) \tag{2.8}$$

$$\mathbf{P}_k(t_i^+) = \mathbf{P}_k(t_i^-) - \mathbf{K}_k(t_i)\mathbf{H}_k\mathbf{P}_k(t_i^-) \tag{2.9}$$

where the $-$ and $+$ superscripts indicate before and after the measurement is taken, respectively, and the subscript k denotes the k^{th} elemental filter in the filter bank. Note that the residuals can be shown to be zero mean, and with the filter-predicted covariance of the k^{th} filter's residuals described by $\mathbf{A}_k(t_i)$, as computed in Equation (2.5).

The state estimates, $\hat{\mathbf{x}}$, are optimal provided that the filter model matches the true characteristics of the airframe. In the multiple model structure, each elemental filter is based on a different failure hypothesis, so this relationship can hold true for at most one filter (and then only if the parameters can assume only the hypothesized point values). This filter will then have the most accurate estimates, and the correct hypothesis is easily selected by monitoring the residuals in the K filters. Consider, however, the general situation in which the true failure mode does not match any of the discretized parameter point values. In this case, the probability evaluator is called on to formulate a best guess based on the residuals from all of the filters.

2.3.2 Conditional Hypothesis Probability Evaluator. While the state estimates from the filters are passed on to the corresponding elemental controller, the residuals are evaluated by the conditional hypothesis probability evaluator. As the name implies, this element calculates, for each Kalman filter, the probability that its hypothesis is currently correct, conditioned on knowledge of the measurement history. This probability can be expressed as

$$p_k(t_i) = Prob(\mathbf{a} = \mathbf{a}_k | \mathcal{Z}(t_i) = \mathcal{Z}_i) \tag{2.10}$$

where $\mathcal{Z}(t_i)$ represents the time history of measurements up to and including that taken at time t_i, $\mathbf{a}$ is the random variable representing the failure condition, and $\mathbf{a}_k$ is the realization of the failure assumed by the k^{th} elemental filter.

It has been shown [25] that probabilities can be computed as:

$$p_k(t_i) = \frac{f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1})p_k(t_{i-1})}{\sum_{j=1}^{K} f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathcal{Z}_{i-1})p_j(t_{i-1})} \tag{2.11}$$

Note the function of the denominator; it acts as a scaling factor to ensure that the probability is properly defined in the sense that

$$p_k(t_i) \geq 0 \quad \text{for all} \quad k \quad \text{and} \quad \sum_{k=1}^{K} p_k(t_i) = 1 \tag{2.12}$$

The conditional density function in Equation (2.11) is recognized as the density function for the current measurement based on the assumed parameter value and the observed time history of all previous measurements, and can be expressed as:

$$f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1}) = \beta_k(t_i) e^{[-\frac{1}{2}\mathbf{r}_k(t_i)^T \mathbf{A}_k{}^{-1}(t_i)\mathbf{r}_k(t_i)]} \tag{2.13}$$

where

$$\beta_k(t_i) = \frac{1}{(2\pi)^{\frac{m}{2}} |\mathbf{A}_k(t_i)|^{\frac{1}{2}}} \tag{2.14}$$

Careful inspection reveals that Equation (2.13) is actually the density function for the k^{th} Kalman filter's residuals, which are white and Gaussian with zero mean and covariance $\mathbf{A}_k$.

For convenience, define the quadratic term in Equation (2.13) to be the likelihood quotient:

$$L_k(t_i) = \mathbf{r}_k(t_i)^T \mathbf{A}_k{}^{-1}(t_i)\mathbf{r}_k(t_i) \tag{2.15}$$

Assume that the k^{th} filter's hypothesis best matches the true failure mode. Then its residuals will have a mean squared value most in consonance with the corresponding filter-predicted covariance, and therefore the likelihood quotient will approach m (the dimension of the measurement vector, $\mathbf{z}(t_i)$, and the residual vector, $\mathbf{r}_k(t_i)$). Compare this to the likelihood quotient for the mismatched filters. As the measurements deviate farther from the filter-predicted values, both the residual vector and the likelihood quotient increase in magnitude. Referring back to Equation (2.13), when multiplied by the negative scalar coefficient, the argument of the exponential term becomes increasingly negative, and so the density calculations for the mismatched filters quickly drop off toward zero. On the other hand, the terms corresponding to the best matched filters approach a finite positive value, $e^{-m/2}$. The terms corresponding to the more closely matched filters will then dominate when the probability is calculated using Equation (2.11). The ability of the MMAC to converge to the correct hypothesis is critical to its operation. Although convergence can be guaranteed theoretically, the necessary proofs [7, 14, 37] provide no insight as to the time involved. Fortunately, empirical studies of failures in flight control systems have shown that the filters do indeed converge in a reasonable amount of time (on the order of seconds or fractions of seconds).

Having determined the corresponding probability for each filter's hypothesis, a decision must be made in order to determine which hypothesis will be considered correct, and how the control law will be generated. One approach, the maximum a posteriori (MAP) approach, is to believe the hypothesis with the highest corresponding probability. Previous research has shown that this decision logic results in quick convergence to the single filter that has the smallest residuals. However by converging to a single filter, the MAP method precludes the option that the true parameter realization may in fact fall outside of the discretized parameter space. For example, an MAP implementation confronted with a partial rudder failure might converge to the hypothesis of a complete rudder failure. The same partial rudder failure would be better modeled as a blending of estimates from two filters based on the fully functional hypothesis and the completely failed rudder hypothesis. Even better, would be the inclusion of a new point in the discretized parameter space which corresponds to the partial failure. However, the number of points needed to cover all partial failures grows quickly and without bound, so the blending is the best solution for implementation. The Bayesian approach accomplishes the required blending by using a weighted average to construct the state estimate (for an MMAE) or the control law (for an MMAC).

The ability of Bayesian blending to compensate for any realization of **a** far outweighs any loss of speed of convergence, and empirical results have shown that very little speed of convergence is lost in practice anyway, so the Bayesian method is used in this research. It should be noted however, that other options for selecting the control law do exist, including a combination of the maximum likelihood and the Bayesian technique. This issue will be discussed further in Section 2.4.1 when modifications are made for implementation.

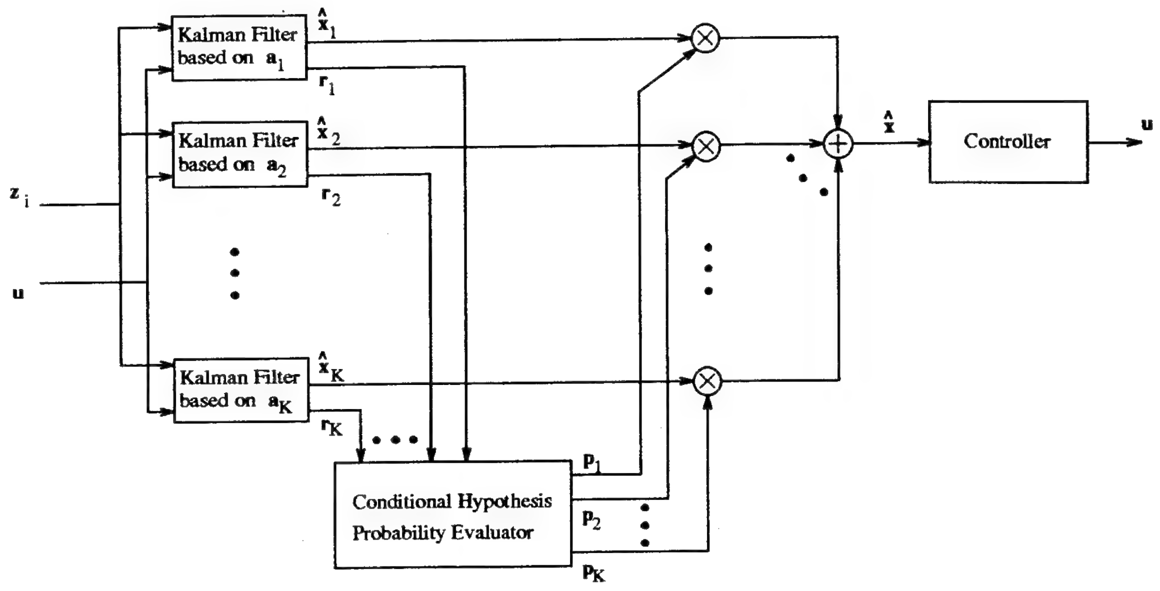


Figure 2.2 Multiple Model Adaptive Estimation Based Control With State Estimation

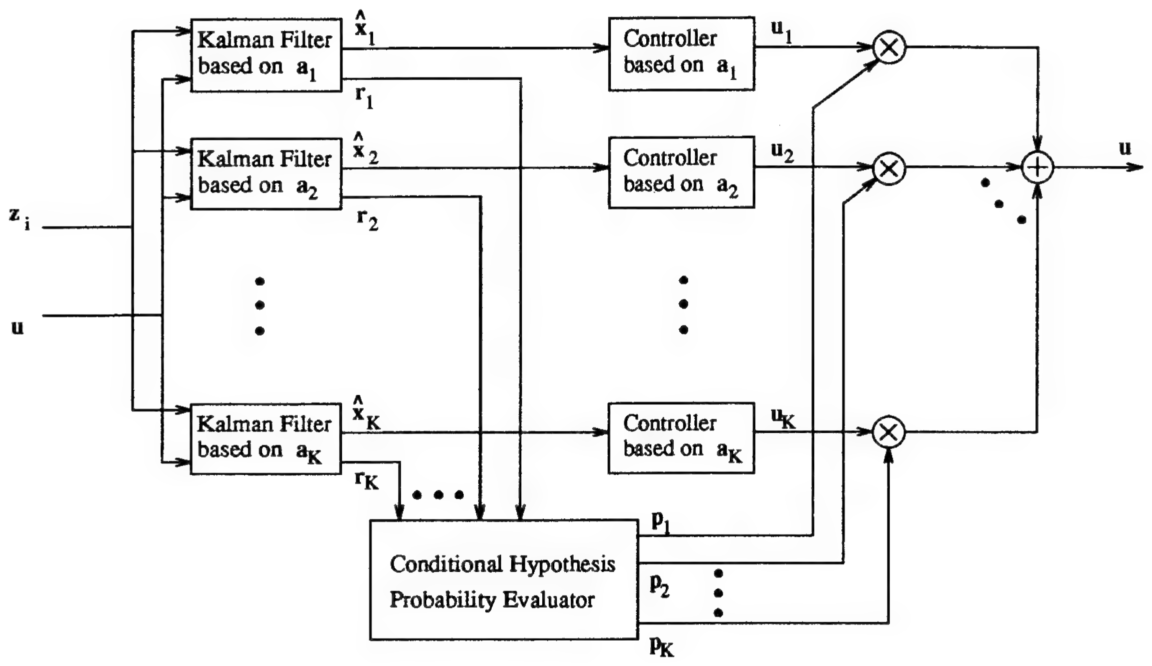
2.3.3 MMAE-Based Control vs. MMAC. There are several ways to provide control within a multiple model framework. One method is to use MMAE-based control, shown in Figure 2.2, which is constructed by sending a probability-weighted state estimate in the form of

$$\hat{\mathbf{x}}_{MMAC}(t_i^+) = \sum_{k=1}^{K} p_k(t_i)\hat{\mathbf{x}}_k(t_i^+) \tag{2.16}$$

to a single robust controller. Similarly, the single controller may not require full-state feedback, but instead rely on a set of measurements (possibly just a subset of the states), estimates of which can be calculated from the state estimates by

$$\hat{\mathbf{z}}(t_i^+) = \mathbf{H}\hat{\mathbf{x}}_{MMAC}(t_i^+) \tag{2.17}$$

MMAE-based controllers providing state or measurement estimates have already been implemented for the LAMBDA URV [18,26] and the VISTA F-16 [8,35,48].



Kalman Filter based on $\mathbf{a}_1$
Kalman Filter based on $\mathbf{a}_2$
Kalman Filter based on $\mathbf{a}_K$
$\hat{\mathbf{x}}_1$
$\mathbf{r}_1$
$\hat{\mathbf{x}}_2$
$\mathbf{r}_2$
$\hat{\mathbf{x}}_K$
$\mathbf{r}_K$
$\mathbf{z}_i$
$\mathbf{u}$
Controller based on $\mathbf{a}_1$
Controller based on $\mathbf{a}_2$
Controller based on $\mathbf{a}_K$
$\mathbf{u}_1$
$\mathbf{u}_2$
$\mathbf{u}_K$
$\mathbf{u}$
Conditional Hypothesis Probability Evaluator
p_1
p_2
p_K

Figure 2.3 Multiple Model Adaptive Controller

An MMAC, by incorporating a bank of elemental controllers specifically tuned for the desired failure conditions, has the ability to provide greater performance than the above MMAE-based control. For instance, in this specific application, both types of controllers can adapt appropriately to sensor failures, but only the MMAC can reconfigure the control law to provide the best possible control in the presence of actuator failures. The output of the MMAC, depicted in Figure 2.3, is the probability weighted control law:

$$\mathbf{u}_{MMAC}(t_i) = \sum_{k=1}^{K} p_k(t_i)\mathbf{u}_k \tag{2.18}$$

Alternatively, the MMAE-based controller can also directly provide parameter estimates, $\hat{\mathbf{a}}$, to be used in the calculation or scheduling of the controller in addition to the state or measurement estimates. This form of MMAE-based control, shown in Figure 2.4, has, like the MMAC, the advantage that the controller is explicitly made aware of change in parameters and can adapt accordingly. Both MMAC and MMAE-based implementations will be investigated in this thesis using LQG controller and control redistribution, respectively.

2.3.4 LQG Elemental controllers. The elemental controllers of Figure 2.3 can be designed using any synthesis technique, the only criteria being that the resulting MMAC converge quickly and be robust enough to provide a stable response during the adaptation process. An LQ design, based on the results of past research [20, 29, 31], satisfies these requirements. Additionally, it is

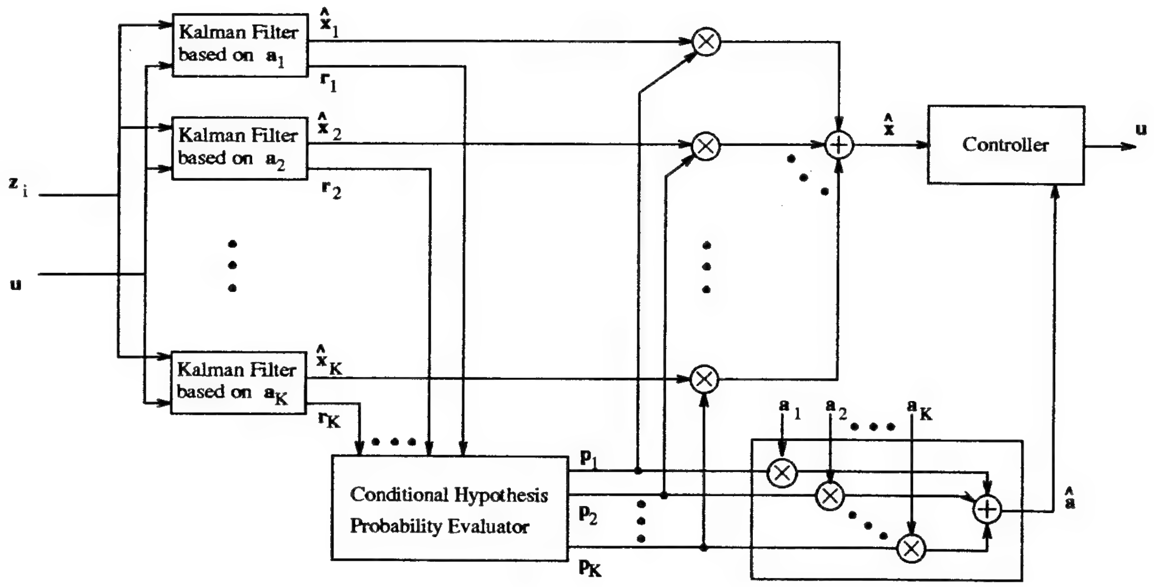


Figure 2.4 Multiple Model Adaptive Estimation Based Control With Parameter Estimation

well suited for use in multivariate control. Equally important is the separation property [24] which applies to Kalman filters and LQG controllers. This principle allows a deterministic LQ controller to be designed based on the assumption of full-state feedback, $\mathbf{x}$, with the actual state feedback being replaced by an estimate, $\hat{\mathbf{x}}$, in implementation. Because of the separation property, this substitution yields an optimally performing stochastic controller. This attribute allows direct use of the Kalman filter structure already designed for the VISTA F-16 MMAE-based controller. The LQG design process is presented more thoroughly in Chapter 3.

Not only can the elemental controllers be designed via any synthesis technique, but, in fact, the MMAC has no requirement that all controllers be synthesized by the same method. Therefore, in order to maintain the integrity of the VISTA F-16 to the fullest extent, the existing Block 40 flight control system is used wherever possible. For example, the Kalman filter based on the fully functional hypothesis will be linked to the Block 40 controller. Also, since the Kalman filters already compensate for missing sensor information, the Block 40 controller will also be linked to elemental filters hypothesizing sensor failures. For single actuator failure hypothesis and for dual failure hypotheses involving an actuator and a sensor, the LQG controller having the corresponding actuator failure hypothesis embedded in its design is used. The decision to use links to the controllers (i. e. have one Block 40 controller that gets called several times for different hypotheses), versus running multiple copies of the same controller, was made after optimized simulations, executing controllers sequentially, indicated that real-time operation may be possible without the need for parallel processors.

2.3.5 Reverse Engineering the Block 40 Flight Control System. The first step in the LQG synthesis will be to reverse engineer the Block 40 flight control system by designing an LQG controller with the same structure and performance characteristics. Although an LQG controller will not be used for the fully functional hypothesis, there exist two motivations for designing this as the initial LQG controller. First, insights gained from this first design, particularly with relation to the weighting matrices, will be exploited in the synthesis of LQG controllers for the failed actuator hypotheses. Second, by designing the controller's response to be similar to that of the Block 40, the LQG controller should meet or exceed military aircraft specifications as given in MIL-STD-1797A [36]. In turn, since the LQG controllers for failed hypothesis should have response characteristics close to this initial design, they too should be as close to meeting specs as can be reasonably expected for a failed aircraft.

Because of the nonlinear nature of the Block 40, it will not be possible to match its structure completely with a linear synthesis technique such as the LQG synthesis. However, certain insights as to the structure can still be gained. For instance, it is clear on the Block 40 functional block diagram [11] that the longitudinal channel relies on a proportional plus integral (PI) control to provide tracking. Therefore, PI control will also be incorporated into the LQG synthesis. Because of the inability to match the nonlinear and linear structures, the performance characteristics will be used as the primary basis for comparison. The quadratic weights used in the LQG synthesis will be chosen such that the resulting controller, when simulated on the nonlinear truth model, approximates the responses given by the Block 40 simulated on the same truth model.

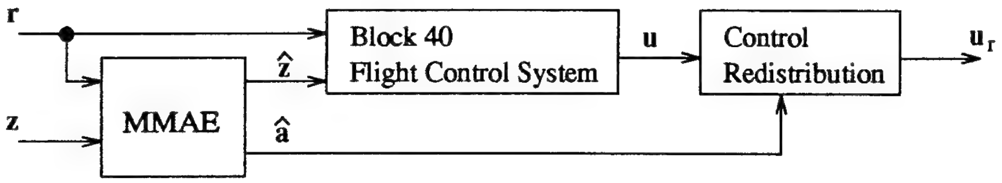


Figure 2.5 Block 40 with Control Redistribution

2.3.6 Alternate Controller Synthesis: Control Redistribution. The MMAE-based controller has the advantage that the existing Block 40 flight control system is carried over with no internal modifications. The only enhancements, as shown in Figure 2.5, are an MMAE front-end which provides state estimates, and control redistribution logic which, based on a parameter estimate from the MMAE, reroutes commands intended for a failed actuator to other functional actuators that can, in combination, have the same effect on the aircraft as the actuator that is no longer available due to failure. If the estimated parameter corresponds to either a fully functional aircraft or sensor failures, then the control signals are simply passed through, and the effective system is identical to a simple MMAE-based controller using only the Block 40 controller. As with the LQG synthesis, the development of the control redistribution will be developed more thoroughly in Chapter 3.

2.4 Modifications Necessary for Implementation

As presented, the basic multiple model theory is complete. However, many modifications and enhancements are made before the design is implemented. With one exception (the second lower bound described in Section 2.4.1 is only applicable to an MMAC), these enhancements apply equally to MMAC and MMAE-based controllers. Therefore, for the remainder of this thesis, the term MMAC will be used generically to refer to both control methodologies, except where obvious by content. The remainder of this chapter will summarize the necessary modifications.

2.4.1 Lower Bounding. Due to the recursive nature of the probability calculation, Equation (2.11), if a probability is ever allowed to become zero, it will remain at zero thereafter, effectively locking out the corresponding hypothesis. To prevent this from occurring, the probabilities are artificially lifted off of zero by introducing an artificial lower bound [2], and then rescaling the probabilities so that they still sum to one. For the VISTA F-16 MMAE implementation where the output is only a state estimate, the lower bound was selected with no regard for the control generation to be 0.001 [31]. The larger the magnitude of this lower bound, the faster probabilities will respond to an actual failure. However, when used with Bayesian blending, large magnitudes attributed to unlikely hypotheses also result in degraded performance due to undesirable control efforts being summed into the MMAC control law via Equation (2.18). To maintain performance, a second lower bound (of larger magnitude) is then introduced, and the probabilities that fall below

Table 2.1 Summary of Lower Bounds

	Effect on...	
	Probability Calculation	Control Law Computation
$p_k > 0.003$	No change	No change
$0.003 > p_k > 0.001$	No change	$\mathbf{u}_k$ set to $\mathbf{0}$
$p_k < 0.001$	p_k set to 0.001	$\mathbf{u}_k$ set to $\mathbf{0}$

this bound are set to zero (and all other probabilities rescaled so as to still sum to one) before being included in control vector calculation of Equation (2.18). Note that these modified probabilities are not used in Equation (2.11), so that the second lower bound has no impact on the probability calculations. Previous empirical research for flight control applications placed this bound at 0.003 [8]. When one or both lower bounds are applied, the probability computation method is termed the modified Bayesian approach. The effects of the lower bounds are summarized in Table 2.1.

2.4.2 Beta Dominance. Beta dominance [2] is the tendency of the probability evaluator to calculate probabilities incorrectly because of an erroneous contribution by the β_k term of Equations (2.13) and (2.14). The result is a tendency to declare incorrect failures corresponding to filters with the smallest precomputed covariance, $\mathbf{A}_k$. Consider the situation where two filters have nearly the same value for the likelihood quotient, $L_1 = L_2 = L$. One would expect that the MMAC would assign equal probabilities to both filters, and indeed the exponential term of the probability density, Equation (2.13), will have the same value, $e^{-\frac{1}{2}L}$. However, the β_k term will typically differ, based on the magnitude of $|\mathbf{A}_k|$, resulting in the filter with the smaller (in the norm sense) filter-predicted covariance being weighted more highly than the other. Recall from Equation (2.5) that the covariance is calculated as $\mathbf{A}_k(t_i) = \mathbf{H}_k\mathbf{P}_k(t_i^-)\mathbf{H}_k^T + \mathbf{R}_k$. All other things being equal, the filter whose hypothesis predicts a sensor failure will tend to have a smaller filter-predicted covariance due to the failure model (presented in Section 3.5) which zeros out the row of $\mathbf{H}$ corresponding the failed sensor. Previous multiple model implementations for flight control systems have indeed demonstrated a propensity for false sensor failure declarations.

Two methods have been used to remove the beta dominance effect. The first uses scalar residual monitoring [31], discussed in Section 2.4.7, to provide additional votes to determine whether or not a sensor failure declaration is correct. While demonstrated effective at eliminating false alarms [31,35], this method does not prevent beta dominance from adversely affecting the MMAC by introducing erroneously high probabilities. A second technique simply removes the β_k term altogether [22, 31]. This technique has demonstrated even better results than that of residual monitoring [8, 26], and will be used in this research. Even though the resulting expression,

$$f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1}) = e^{[-\frac{1}{2}L_k(t_i)]} \tag{2.19}$$

is no longer a proper Gaussian density (the area under the function is no longer unity,) this is of no overriding concern since the density function is used only to calculate the p_k probabilities in Equation (2.11). The normalizing effect of the denominator in Equation (2.11) assures that use of Equation (2.19) still yields a valid set of probabilities in the sense that Equation (2.12) is satisfied.

2.4.3 Scalar Penalty. The scalar coefficient of $-\frac{1}{2}$ in the exponential term of Equation (2.13) is directly related to the sensitivity of the MMAC [26]. Increasing the magnitude of this term drives probabilities associated with large residuals to zero faster. Ideally, this should result in a faster adaptation process; however, increasing the scalar penalty also increases the incidence of false alarms. The $-\frac{1}{2}$ coefficient can then be treated as a sensitivity gain which can be adjusted to tune performance if need be.

2.4.4 Dither. Dither is the introduction of low magnitude control signals in order to excite the system and enhance identifiability. To illustrate the need for a dither signal, consider an aircraft flying in a straight and level flight path. Should a rudder fail, the loss or degradation in control authority may well go unnoticed by both the pilot and the adaptive controller, particularly if the surface fails in such a manner that it floats in the relative wind (a failure to free stream).

Only after the pilot initiates a maneuver will the residuals in the MMAC begin to reflect the change in system parameters, resulting in a period of degraded performance until the failure is declared. A better design detects the failure when it happens regardless of flight path, allowing the MMAC to reconfigure appropriately and provide maximum performance to the pilot at all times.

Subliminal dither signals force this early detection by exercising the control surfaces, which excites the state vector. Any deviation from the filter-predicted state values will then show up in the residuals and trigger a failure declaration. Though several forms of dither have been suggested [18, 35, 48], varying waveform, frequency content, and amplitude. arguably the best for enhancing failure detection is a simple sinusoid [35]. A periodic dither has the additional advantage that sophisticated algorithms can monitor the residuals not just for magnitude but also for frequency effects corresponding to the input dither. By sending a different frequency or phase to each channel, the confounding effects of cross-coupling between channels can be minimized. The dither signals need to be kept subliminal to reduce pilot strain, but with an associated drawback that the smaller amplitudes required for subliminal dither are less effective at enhancing parameter identification. An alternative implementation may be to introduce larger magnitudes, but infrequently and only for short periods of time, possibly even allowing the pilot to control their application. Note that dither is only needed during benign flight conditions when the MMAC does not receive enough information to make accurate decisions. During strong maneuvers, the system is excited sufficiently without the addition of dithering. In fact, dither may not be needed at all if pilots were trained to induce maneuvers which shook up the system either periodically throughout the flight or just before the onset of deliberate maneuvers.

Of utmost concern is the need to keep the dither subliminal so as not to introduce additional discomfort or fatigue on the pilot. The effects of vibration on human subjects was researched in the 1960's in conjunction with the space program [12, 45]. Although scientists at the time were considering unwanted vibrations, their conclusions are equally pertinent to dither, which can be thought of as an induced vibration. The limits on dither magnitude used for the VISTA F-16 stem from this research, and the maximum allowable magnitude response is set to be $\pm$ 0.1 g's in the longitudinal channel and $\pm$ 0.2 g's in the lateral channel.

The initial dither signal used in this research is chosen based on past work [35] to be sinusoidal with a frequency 15 radians/second. The signal is applied to each of the three control channels (pitch stick, roll stick, and pedals) at an appropriate amplitude to be considered subliminal. Section 4.5 details changes that were made to the dither signal before successful detection of all failure conditions was possible.

2.4.5 Hierarchical Structure. The VISTA F-16 flight control system runs at 64 Hz, requiring all computations for the filters, controllers, and probability evaluator to take place in less than 16 msec. Clearly the computational loading of the MMAC is a severe constraint, especially as the number of points in the discretized space grows. For example, if only single failure hypotheses are considered, $1+K$ filters and controllers are required (one fully functional hypothesis and K different failure hypotheses). However, if both single and double failure are allowed, the number grows to $1+K+\frac{K!}{(K-2)!2!}$. In this research, where eleven failure conditions are considered, these numbers are 12 and 67, respectively. If only single failures were considered, a parallel implementation of twelve processors could be used to bring the effective computational loading down to that of one filter, one controller, and the probability evaluator. However, in the case of single and double failures, a real time implementation would require 67 parallel processors - a very poor allocation of computer resources.

The solution is to implement the hierarchical structure [31] depicted in Figure 2.6. The failure hypotheses are grouped in banks of twelve, and only one bank is put on-line at once. Should a failure be detected in the k^{th} filter, a failure being defined as having the corresponding probability, p_k, be over a given threshold for a given number of samples, the "Level 0" bank is switched out for the "Level 1" bank that hypothesizes two failures: the k^{th} failure and an additional second

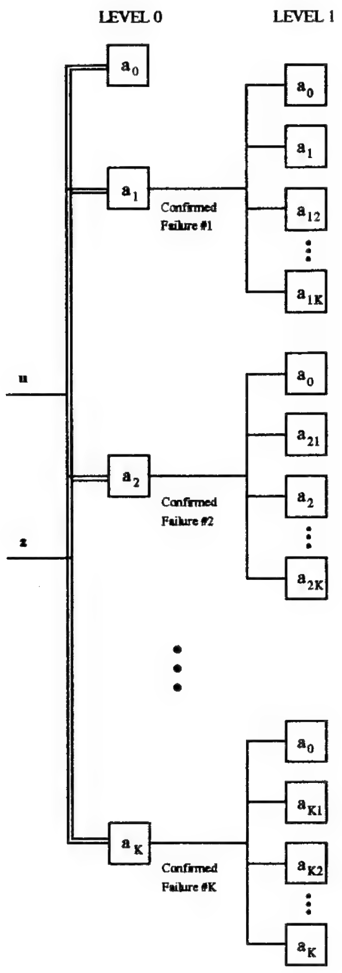


Figure 2.6 Hierarchical Structure

failure. To allow for the possibility that a false initial declaration is made, each "Level 1" bank also contains a filter hypothesizing that no failure occurred, allowing the MMAC effectively to back out of the false declaration by moving back up to the "Level 0" bank. The depth of the hierarchical structure can be extended as needed, with the MMAC able to traverse up and down between banks at different levels, but never between two banks at the same level.

For the MMAC used in this research, the probability threshold is set at 0.95, and the time interval over which it must be exceeded is set to be one sample period. With failure modes confined to single or double failures only, this MMAC only requires a two-level hierarchical structure, with each bank containing eleven failure modes and one fully functional mode.

2.4.6 Probability Smoothing. Immediately following a failure insertion into the truth model, the probabilities go through a transient period before converging to the correct solution. Probability smoothing [26] is introduced to reduce the possibility that the MMAC will declare false alarms based solely on these transients. Removing these false alarms becomes even more important when using a hierarchical structure since a failure declaration results in bank swapping. The probabilities are smoothed using a moving window of a given size, which is specific to each application. The MMAE-based controller implementation for the VISTA F-16 [8] did not require a probability smoothing at all (a window size of zero). Note that smoothed probabilities are used only for making the decision as to whether or not a failure has occurred and to invoke bank swapping in the hierarchical structure; the blended control law calculated in Equation (2.18) uses the unsmoothed probabilities.

2.4.7 Scalar Residual Monitoring. The MMAC calculates probabilities based on the likelihood quotient, $L_k(t_i)$. Though unused in this MMAC, additional insight can be gained by monitoring the scalar residuals themselves [31, 35]. Referring to the residuals from the filter hypothesizing a fully functional aircraft, a sensor failure will manifest itself not only as an increase in the likelihood quotient, but also as a direct increase in one of the corresponding scalar residuals. Checking these terms can then be used as an additional vote either to increase the convergence time if there is an indication of convergence to the wrong hypothesis, or to eliminate false alarms. For example, when beta dominance causes false alarms in the filters hypothesizing a yaw sensor failure, the MMAC can test the scalar residual corresponding to yaw. If the result of the test does not corroborate the failure declaration, then the failure declaration is labeled false and is ignored.

Several methods exist for testing scalar residuals, all of which rely on the statistical properties of residuals, which are white Gaussian processes with zero mean and a variance of the appropriate diagonal term of $\mathbf{A}_k(t_i)$, provided that the presumed k^{th} hypothesis is correct. A simple way to evaluate the whiteness of the residual is to count the number of zero crossings [35], once the residuals are corrected for zero-meanness. A white process is indicated by a significantly higher count than would be present for colored noise. In practice, an arbitrary threshold can be set, above which whiteness is declared satisfied. Another test counts the number of times that the residual breaks the $\pm\sigma$ boundaries [25]. For a true Gaussian distribution, the residual should be inside the one-σ boundary 68.3% of the time. Similarly, statistics for the two- and three-σ boundaries are 95.4% and 99.7% respectively. Because failures will often appear as a bias in the residuals, a third test is to calculate the mean of a finite number of samples. Any deviation from a small range about zero could then indicate, or be used to corroborate, a failure.

Other more sophisticated tests also exist, such as calculating the scalar quadratic term $r_{kj}^2/A_{k_{jj}}$, representing the ratio of the true square of the j^{th} scalar residual to its filter predicted variance in the k^{th} elemental filter [35]. Similar to the likelihood quotient calculation, filters with the correct hypothesis will have residual-squared values most in consonance with the filter's predicted variance. This method is particularly useful for detecting sensor failures which tend to affect the single scalar residual corresponding to the associated measurement, unlike actuator failures which appear in multiple scalar residuals (and are thus harder to detect in general for any method). Also, if dither is applied, the appearance of that dither frequency in a given residual is another indication of incorrectness of the corresponding hypothesis [35]. The dither, which is very nonwhite and easily detected since it is of known frequency, should not appear in the otherwise zero-mean, white residuals if the corresponding hypothesis is correct. In this research, scalar residual monitoring will not be used unless false alarms or poor convergence dictate the need for corroborating votes.

2.5 Chapter Summary

This chapter highlighted the historical development of multiple model techniques, particularly at AFIT where a strong sequence of applications based on the F-15, the LAMBDA URV, and the VISTA F-16 has created a solid foundation for continued research into multiple model adaptive control. A brief overview of the MMAC was then provided, accompanied by a detailed description of the three basic elements: the Kalman filter, the LQ full-state feedback controller, and the probability evaluator. A comparison of MMAC and MMAE-based control was made, and control redistribution was introduced as a viable alternative to a true MMAC implementation. Finally, the modifications necessary to implement a useful MMAC were given, including lower bounding, removing beta dominance effects, and applying a subliminal dither. The next chapter will focus more on the design and implementation of the controllers.

3. Controller Development

3.1 Chapter Overview

This chapter begins with an introduction to the host aircraft, the VISTA F-16, depicted in Figure 3.1, and a description of the available simulation. Sections 3.3 and 3.4 present the truth and design models used for controller synthesis. Immediately following is a description of the failure models used for both actuator and sensor failures. Section 3.6 then presents the linear quadratic full-state feedback controller in its entirety. The need for proportional plus integral control is addressed in Section 3.6.2, including explicit details on how to implement PI control within the framework of a linear quadratic regulator synthesis. Other enhancements, such as implicit and explicit model following, are mentioned, followed by a step-by-step summary of the LQG design procedure so that the interested reader can reproduce the results. Section 3.8 develops the control redistribution method used for the MMAE-based controller. Finally, a chapter overview is given in Section 3.9.

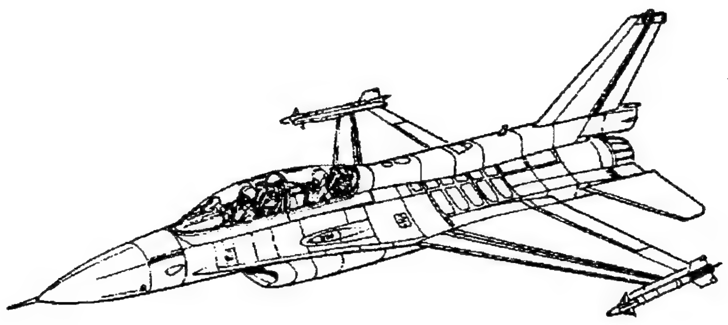


Figure 3.1 VISTA F-16

3.2 VISTA F-16

Designed to replace the aging NT-33A, the VISTA F-16 is a Variable-Stability In-Flight Simulator Test Aircraft which provides an in-flight simulator for modern, high-performance aircraft. Through the variable stability flight control system (VSS), the VISTA F-16 is able to provide test pilots with the look and feel of another aircraft. The VISTA F-16 is modified from a F-16D airframe and is built under contract by Calspan and General Dynamics [50].

The VISTA F-16 was chosen as the host for a sequence of research at AFIT primarily due to the availability of an advanced flight simulation at the Flight Dynamics Directorate of Wright Laboratory. This simulation, running as part of the Simulation Rapid-Prototyping Facility (SRF) software package, incorporates General Dynamic's VISTA F-16 simulation software with VSS software provided by Calspan. A convenient user interface is provided by the Transportable Applications Executive (TAE), which, among other functions, allows the user to configure the F-16, select a flight condition, and command pilot inputs in non-real time [50]. For this research, the Fortran source code is modified to accommodate the multiple model architecture.

3.3 Truth Model

The truth model used in this research is the SRF VISTA F-16 simulation. The SRF provides a full six-degree-of-freedom simulation using nonlinear equations of motion, and it incorporates features such as advanced actuator modeling, the complete Block 40 controller, and the Aileron Rudder Interconnect, used for turn coordination. As has been done previously, the SRF VISTA simulation is assumed to be a valid model of the VISTA F-16 since verification is not possible within the bounds of this research effort [8]. The availability of a high order truth model is a significant improvement in failure detection and control via MMAC, since earlier attempts at failure detection were either verified using only linear dynamics [20] or else they modeled only the longitudinal channel [29, 31], and therefore their results were not necessarily indicative of the response of a real aircraft.

The VSS capabilities included in the SRF VISTA simulation will not be used in this research. Additionally, the SRF VISTA is modified in the following ways: the existing wind model is replaced by a more accurate model based on on a zero-order Dryden wind model [8, 42]; the limited failure modes are replaced by code which allows dual simultaneous failures at user-specified times [8]; sensor noise is incorporated to provide more realistic measurements [8]; and the lateral acceleration measurement computation is replaced by a linear model for reasons discussed in Section 3.4.2. A more detailed summary of changes made to the original source code can be found in Appendix C.

3.4 Design Model

The design model is selected to be a linear time-invariant, discrete-time model, implemented on the VISTA F-16's digital flight control system. The basis for the design model is the continuous-time model, provided by the SRF VISTA, of the form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3.1}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ X'_\theta & X'_u & X'_\alpha & X'_q & 0 & 0 & 0 & 0 \\ Z'_\theta & Z'_u & Z'_\alpha & Z'_q & 0 & 0 & 0 & 0 \\ M'_\theta & M'_u & M'_\alpha & M'_q & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \phi'_r \\ 0 & 0 & 0 & 0 & Y'_\phi & Y'_\beta & Y'_p & Y'_r \\ 0 & 0 & 0 & 0 & 0 & L'_\beta & L'_p & L'_r \\ 0 & 0 & 0 & 0 & 0 & N'_\beta & N'_p & N'_r \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ X'_{\delta_e} & 0 & X'_{\delta_f} & 0 & 0 \\ Z'_{\delta_e} & 0 & Z'_{\delta_f} & 0 & 0 \\ M'_{\delta_e} & 0 & M'_{\delta_f} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & Y'_{\delta_d t} & 0 & Y'_{\delta_a} & Y'_{\delta_r} \\ 0 & L'_{\delta_d t} & 0 & L'_{\delta_a} & L'_{\delta_r} \\ 0 & N'_{\delta_d t} & 0 & N'_{\delta_a} & N'_{\delta_r} \end{bmatrix}$$

The state and input vectors are defined in Table 3.1.

Table 3.1 State and Input Vectors

x		State Variables	Units
1	θ	Pitch angle	rad
2	u	Forward velocity	ft/sec
3	α	Angle of attack	rad
4	q	Pitch rate	rad/sec
5	ϕ	Bank angle	rad
6	β	Sideslip angle	rad
7	p	Roll rate	rad/sec
8	r	Yaw rate	rad/sec

u		Input Variables	Units
1	δ_e	Elevator position	rad
2	δ_{dt}	Differential tail position	rad
3	δ_f	Flap position	rad
4	δ_a	Aileron position	rad
5	δ_r	Rudder position	rad

This model facilitates the synthesis and analysis of linear estimators and control laws which minimize the computational burden compared to a design for the nonlinear truth model. However, before implementation, the input matrix will be redefined, measurement and controlled variable output models added, white noise incorporated into the dynamics and measurement models, and finally the system converted to a discrete-time system. These modifications are discussed in the remainder of this section.

3.4.1 Analysis of the Linear Model. The plant matrix, **A**, is expressed in terms of primed dimensional stability derivatives (primed indicating that angles, α and β, are included in the state vector instead of velocities, v and w) which can be supplied as an option of the SRF VISTA simulation. First, the aircraft configuration is specified via the TAE as "up-and-away" flight with two AIM-9L missiles and wing tanks empty. Then the flight condition, chosen to allow verification with the MMAE work already accomplished [8], is selected to be Mach 0.4 at an altitude of 20,000 feet in a standard climate (purposefully chosen to be a challenging condition with low dynamic pressure). The resulting primed stability derivatives, grouped in terms of forces (X,Y,Z) and moments (L,M,N) about the body axis, are given in Table 3.2. Complete descriptions of the individual stability derivatives and the body axis are given by Blakelock [3] and Nelson [38].

Table 3.2 Primed Dimensional Stability Derivatives

X'_θ = -31.6771 (ft/sec^2)	Z'_θ = -.141578E-01 $(1/sec)$	M'_θ = 0.642121E-03 $(1/sec)$
X'_u = 0.243115E-02 $(1/sec)$	Z'_u = -0.195975E-03 $(1/ft)$	M'_u = -0.134645E-02 $(1/ft \cdot sec)$
X'_α= 16.3530 (ft/sec^2)	Z'_α = -0.440414 $(1/sec)$	M'_α = 1.52512 $(1/sec^2)$
X'_q = -73.4589 (ft/sec)	Z'_q = 0.997196	M'_q = -0.526916 $(1/sec)$
X'_{δ_e} = 2.08784 (ft/sec^2)	Z'_{δ_e} = -0.684394E-01 $(1/sec)$	M'_{δ_e} = -3.64478 $(1/sec^2)$
X'_{δ_f} = -0.542584 (ft/sec^2)	Z'_{δ_f} = -0.337867E-01 $(1/sec)$	M'_{δ_f} = 0.285675 $(1/sec^2)$
Y'_ϕ = 0.775991E-01 $(1/sec)$		ϕ'_r = 0.182448
Y'_β = -0.109880 $(1/sec)$	L'_β = -18.5276 $(1/sec)$	N'_β = 2.83301 $(1/sec^2)$
Y'_p = 0.180844	L'_p = -1.55588 $(1/sec)$	N'_p = -0.432911E-01 $(1/sec)$
Y'_r = -0.997627	L'_r = 0.413477E-01 $(1/sec)$	N'_r = -0.282169 $(1/sec)$
$Y'_{\delta_{dt}}$= 0.137713E-01 $(1/sec)$	$L'_{\delta_{dt}}$ = -8.99043 $(1/sec^2)$	$N'_{\delta_{dt}}$= -1.03452 $(1/sec^2)$
Y'_{δ_a} = 0.569506E-03 $(1/sec)$	L'_{δ_a} = -12.4072 $(1/sec^2)$	N'_{δ_a} = -0.130713 $(1/sec^2)$
Y'_{δ_r} = 0.169586E-01 $(1/sec)$	L'_{δ_r} = 2.86294 $(1/sec^2)$	N'_{δ_r} = -1.16519 $(1/sec^2)$

An analysis of the resulting design point shows that the trimmed aircraft is not in true steady-state flight; normal acceleration is trimmed at a non-zero value, $a_n = -0.120997 \times 10^{-3}$. For this research, which only considers eight seconds of flight, a_n is assumed to be negligible, and in fact Figure 3.2 shows that small angles are maintained, so this assumption is justified. Note also the dimensional units of the state and control variables. Although the original SRF output file assigns

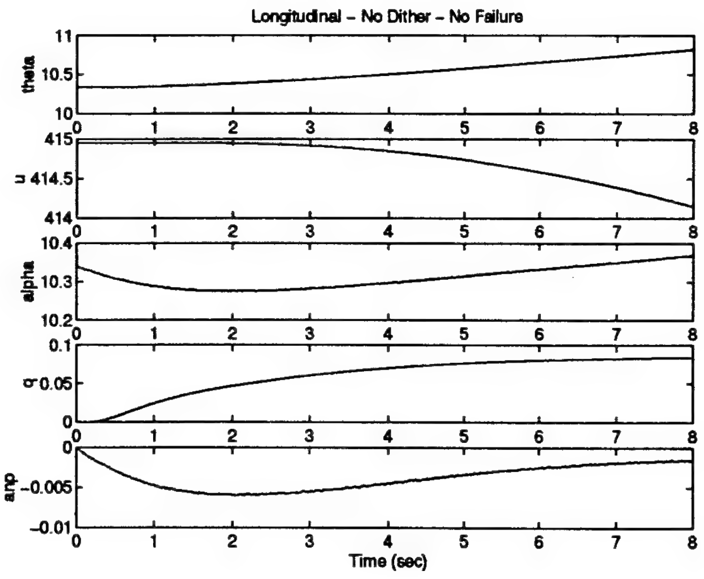


Figure 3.2 Effect of Non-Zero Normal Acceleration Trim

units of degrees to them, a thorough analysis of the source code (particularly of file *convert.F*) shows that the proper set of units for all rotational variables is in fact radians.

The input vector, **u**, merits explanation. The VISTA F-16 has four sets of control surfaces: leading edge flaps, flaperons, stabilators, and a rudder. Note that the linearized model does not include the leading edge flaps in the input vector. These flaps are primarily used for take-off and landing, and have virtually no control authority at the design point used in this research, so their omission is justified. Flaperons span much of the trailing edge of the wings and can be commanded differentially as ailerons to produce rolling moments, symmetrically as flaps to produce pitching moments, or as a blending of the two commands. Stabilators compose the horizontal tail, and they can also be commanded in the same manner as with the flaperons. The VISTA F-16 has a single vertical tail with a rudder on the trailing edge which is used to produce yawing moments and coordination.

For better failure modeling, and to facilitate control of a failed aircraft, the input is rearranged somewhat to provide individual control of each surface position. The modified input vector is given in Table 3.3 and the modified input matrix is:

$$\mathbf{B}_{mod} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}X'_{\delta_e} & \frac{1}{2}X'_{\delta_e} & \frac{1}{2}X'_{\delta_f} & \frac{1}{2}X'_{\delta_f} & 0 \\ \frac{1}{2}Z'_{\delta_e} & \frac{1}{2}Z'_{\delta_e} & \frac{1}{2}Z'_{\delta_f} & \frac{1}{2}Z'_{\delta_f} & 0 \\ \frac{1}{2}M'_{\delta_e} & \frac{1}{2}M'_{\delta_e} & \frac{1}{2}M'_{\delta_f} & \frac{1}{2}M'_{\delta_f} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{2}Y'_{\delta_d t} & \frac{1}{2}Y'_{\delta_d t} & -\frac{1}{2}Y'_{\delta_a} & \frac{1}{2}Y'_{\delta_a} & Y'_{\delta_r} \\ -\frac{1}{2}L'_{\delta_d t} & \frac{1}{2}L'_{\delta_d t} & -\frac{1}{2}L'_{\delta_a} & \frac{1}{2}L'_{\delta_a} & L'_{\delta_r} \\ -\frac{1}{2}N'_{\delta_d t} & \frac{1}{2}N'_{\delta_d t} & -\frac{1}{2}N'_{\delta_a} & \frac{1}{2}N'_{\delta_a} & N'_{\delta_r} \end{bmatrix}$$

Table 3.3 Modified Input Vector

$\mathbf{u}_{mod}$	Modified Input Variables		Units
1	δ_{ls}	Left stabilator position	rad
2	δ_{rs}	Right stabilator position	rad
3	δ_{lf}	Left flaperon position	rad
4	δ_{rf}	Right flaperon position	rad
5	δ_{rud}	Rudder position	rad

3.4.2 Measurement and Output Models. The measurement and controlled variable output models are

$$\mathbf{z}(t_i) = \mathbf{H}\mathbf{x}(t_i) + \mathbf{D}_z\mathbf{u}(t_i) \tag{3.2}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}_y\mathbf{u}(t) \tag{3.3}$$

where the measurement matrices, $\mathbf{H}$ and $\mathbf{D}_z$, and output matrices, $\mathbf{C}$ and $\mathbf{D}_y$, used for estimation and control respectively, are user defined based on the variables of interest. For this research, the measurement variables, shown in Table 3.4, are identical to those used in the previous MMAE implementation with the exception of velocity, u, which is unused by the Block 40 flight control system, and therefore was removed. Note the discrete form of the measurement model in Equation (3.2), reflecting the fact that all sensor readings contain sampled-data measurements. The output variables, also shown in Table 3.4, are chosen to coincide with the control variables specified by the Wright Laboratories Flight Dynamics Directorate for Martin's research [20].

Table 3.4 Measurement and Output Vectors

$\mathbf{z}$	Measurement Variables		Units
1	α	Angle of attack	rad
2	q	Pitch rate	rad/sec
3	a_n	Normal acceleration	g's
4	p	Roll rate	rad/sec
5	r	Yaw rate	rad/sec
6	a_y	Lateral acceleration	g's

$\mathbf{y}$	Output Variables		Units
1	C^*	Blend of q and a_n	rad
2	p	Roll rate	rad/sec
3	β	Sideslip angle	rad

Note: All accelerations are measured at the pilot's station

While most of these variables of interest are readily available as states, models for normal acceleration, a_n, lateral acceleration, a_y, and C^* must be developed. Blakelock [3] gives equations for accelerations at the center of gravity as :

$$\begin{aligned} a_n &= -u(\dot{\alpha} - q) && (3.4) \\ a_y &= u(\dot{\beta} + r) && (3.5) \end{aligned}$$

These accelerations can be translated forward to the pilot's station and converted to the amount of g's (ratio of acceleration caused by aerodynamic forces to the acceleration caused by gravity) felt by the pilot. First, the contribution due to gravity, as resolved into the normal and lateral directions, must be removed so that only aerodynamic forces are considered:

$$\begin{aligned} a_n &= -u(\dot{\alpha} - q) - g\cos(\alpha) && (3.6) \\ a_y &= u(\dot{\beta} + r) - g\sin(\phi) && (3.7) \end{aligned}$$

Next, small angle approximations are used to remove the transcendental terms. Then, the gravity term is divided out so that the units of acceleration are in terms of g's. Finally, because the accelerometers on the VISTA F-16 are located near the pilot's station, the equations must be translated forward from the center of gravity. The resulting models are:

$$\begin{aligned} a_n &= -\frac{u}{g}(\dot{\alpha} - q) + \frac{l_x}{g}\dot{q} && (3.8) \\ a_y &= -\frac{u}{g}(\dot{\beta} + r) - \phi + \frac{l_x}{g}\dot{r} && (3.9) \end{aligned}$$

where g is the acceleration due to gravity $(32.17 ft/sec^2)$ and l_x is the moment arm to the pilots station. Neglecting the small l_z contribution, the SRF VISTA software (source file *afmic.F*) calculates the moment arm as $l_x = 9.988 + \bar{c} \cdot x_{cg} \cdot 0.01$, where $\bar{c}$ is defined as the mean aerodynamic chord and x_{cg} is the location of the center of gravity on the x-axis. Note that for this research, an average value of x_{cg} is used in order to generate filters independent of aircraft configuration. Values used are: $x_{cg} = 37.376$, $\bar{c} = 11.32ft$, and $l_x = 14.219ft$.

C^* is a scheduled blending of pitch rate and normal acceleration based on dynamic pressure used for longitudinal control [41]. The C^* scheduling is modeled as [5]:

$$C^* = g_1 \cdot a_n + g_2 \cos q \tag{3.10}$$

where

$$g_1 = 143/(\bar{q} + 113) \tag{3.11}$$

$$g_2 = (1 - g_1) \tag{3.12}$$

Note that C^* is scheduled based on dynamic pressure, $\bar{q}$, and that at low dynamic pressure C^* is composed mostly of pitch rate, while at high dynamic pressure C^* is predominately normal acceleration. The dynamic pressure corresponding to the flight condition used in this research is 109 lbs/sq ft. This blending, shown graphically in Figure 3.3 with the gains corresponding to 109 lbs/sq ft dynamic pressure marked by an X, was chosen as the longitudinal controlled variable based on the preference of an experienced pilot [41].

Attempts to verify the a_n model show a significant bias, Figure 3.4, which is attributed to the first order approximation used in resolving the gravity contribution in Equation (3.6). Assuming that the angle of attack, α, varies slowly, which it does in this research, there will be no affect on the perturbation states, and the a_n model is acceptable as is. If necessary, however, the bias could also be removed by including the second order term, such that $cos(\alpha) \cong 1 + \frac{\alpha^2}{2}$.

Attempts to verify the a_y model revealed large errors in both magnitude and phase, as seen in Figure 3.5. Since a reasonable explanation is unavailable for this difference, attempts (documented in Section 4.3) were made to incorporate knowledge of the discrepancy. First, the poor sensor reading was acknowledged by adding pseudonoise through increasing the $\mathbf{R}(6,6)$ term, thereby

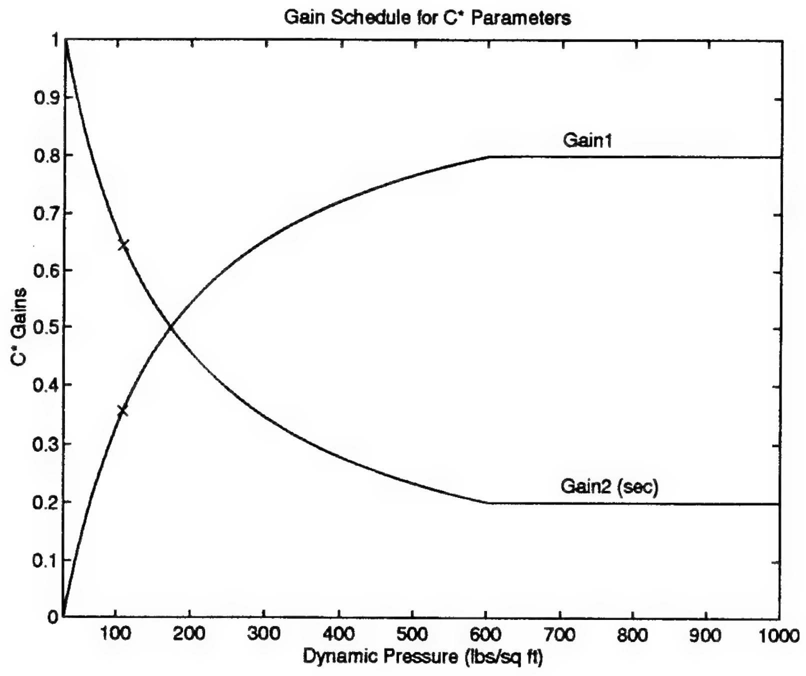


Figure 3.3 C^* Schedule

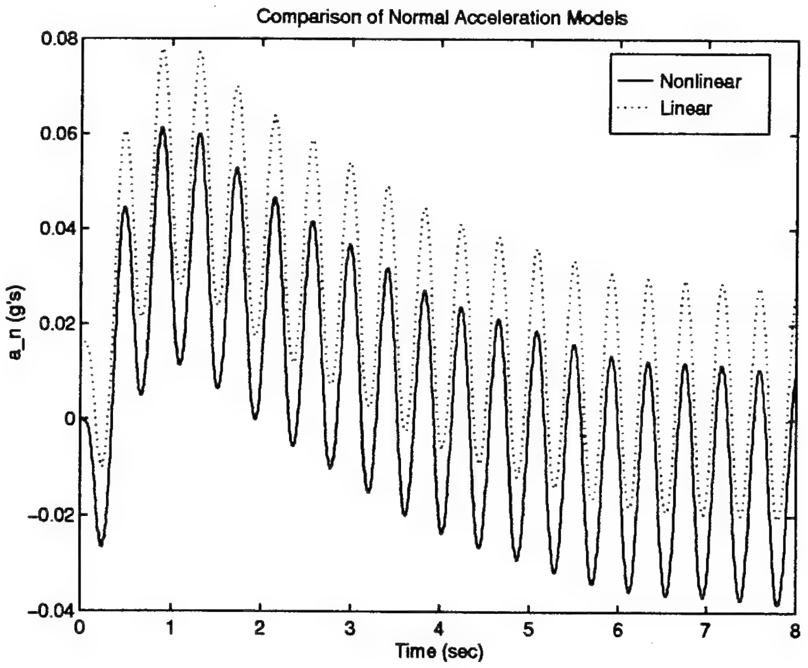


Figure 3.4 Comparison of Normal Acceleration Models

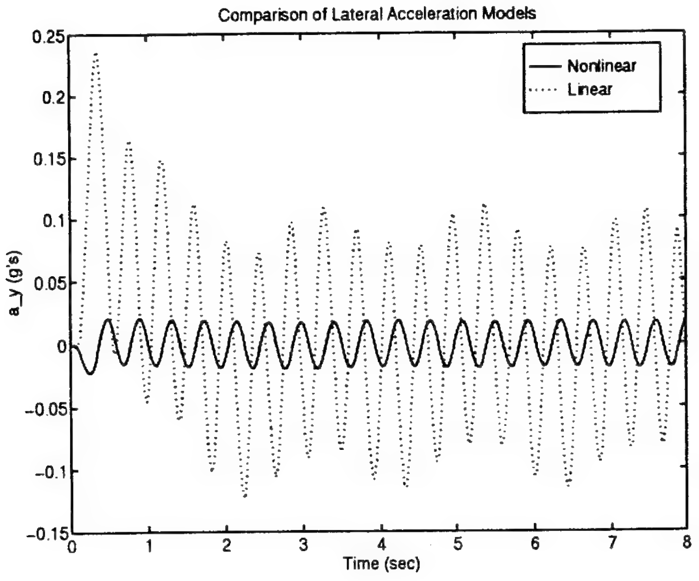


Figure 3.5 Comparison of Lateral Acceleration Models

informing the Kalman filters to rely more on its internal model and less on the sensor reading. When no amount of pseudonoise tuning gave reasonable results, the entire measurement itself was removed. Without the measurement, however, the MMAC was unable to declare failures of the rudder or yaw-rate sensor.

Being forced to include the lateral acceleration measurement, two options were available: form a linear model to match the (possibly incorrect) nonlinear model, or incorporate the linear dynamics into the truth model. Based on time constraints, the latter route was chosen, with Equation (3.7) incorporated as the model for lateral acceleration measured at the center of gravity.

3.4.3 Noise Models. The stochastic nature of problem is incorporated by adding white Gaussian noise to the the plant and measurements. The resulting continuous-time system with sampled-data measurements in state-space notation is:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{Ax}(t) + \mathbf{Bu}(t) + \mathbf{Gw}(t) \\ \mathbf{z}(t_i) &= \mathbf{Hx}(t_i) + \mathbf{D}_z\mathbf{u}(t_i) + \mathbf{v}(t_i) \\ \mathbf{y}(t) &= \mathbf{Cx}(t) + \mathbf{D}_y\mathbf{u}(t)\end{aligned} \tag{3.13}$$

Table 3.5 Plant Dynamics and Sensor Noise

$\mathbf{Q}$	Average Noise Strength		Units
$\mathbf{Q}(1,1)$	u	4.5×10^{-2}	$ft^2/rad \cdot sec$
$\mathbf{Q}(2,2)$	α	3.0×10^{-6}	$rad \cdot sec$
$\mathbf{Q}(2,3)$	α vs. q	1.1×10^{-8}	rad
$\mathbf{Q}(3,3)$	q	1.5×10^{-6}	rad/sec
$\mathbf{Q}(4,4)$	p	6.0×10^{-6}	rad/sec
$\mathbf{Q}(5,5)$	β	3.0×10^{-6}	$rad \cdot sec$
$\mathbf{Q}(5,6)$	β vs. r	6.3×10^{-9}	rad
$\mathbf{Q}(6,6)$	r	2.4×10^{-6}	rad/sec

$\mathbf{R}$	RMS Noise		Units
$\mathbf{R}(1,1)$	α	0.004	rad
$\mathbf{R}(2,2)$	q	0.006	rad/sec
$\mathbf{R}(3,3)$	a_n	0.01	$g's$
$\mathbf{R}(4,4)$	p	0.02	rad/sec
$\mathbf{R}(5,5)$	r	0.006	rad/sec
$\mathbf{R}(6,6)$	a_y	0.005	$g's$

where dynamics driving noise, $\mathbf{w}(t)$, represents noise gusts and mismodeling effects (due to linearization and low-order models) and discrete-time noise, $\mathbf{v}(t_i)$, represents sensor noise. Previously established values for average strength of the dynamics driving noise and RMS values for measurement corruption noises, as given in Table 3.5, are used for the noise models [8,34,42,46]. Note that these tables show the values actually used in the previous research, not necessarily the values that were documented. Lacking data for the VISTA F-16, the sensor model RMS values are conservative estimates based on Elliott's 1977 description of the F-8 digital fly by wire aircraft [9]. Considering the advances in technology over the last 18 years, the VISTA F-16's sensors should be considerably more accurate than those of the F-8, resulting in lower RMS values for sensor models which would greatly enhance the MMAC's ability to detect failures. The bandwidth of these sensors was given to be at least 300 rad/sec. In relation to the low frequency response of the aircraft, the sensor noise can be considered essentially white [20].

Pogoda [42] originally derived the dynamics noise model for the STOL F-15 from the Dryden wind model specified in MIL-STD-1797A [36]. This model was later adapted for use on the VISTA F-16 [8,34]. The zero-order Dryden wind model was also incorporated into the SRF VISTA simulation, which previously had neglected all rotational effects of the Dryden wind model [8]. This improved SRF wind model is retained for this research, and the zero-order Dryden wind model is incorporated into the design model. The derivation of the zero-order Dryden wind model is presented in Appendix B for reference.

Note that the value of the dynamics noise strength for the velocity term (i. e. , the noise directly driving the velocity differential equation in Equation (3.13)), has been decreased by two orders of magnitude to reflect the fact that the original noise models were found to be representative of heavier turbulence than the light turbulence specified [8]. Also, the units of the average noise strength, Table 3.5, are appropriate since the noise injection matrix, **G** in Equation (3.13), is dimensional. Because the units corresponding to direct feed-through of each white noise component are $1/sec$, the appropriate units of the white noise components must match the units of the corresponding state variable. The units of **Q** are then calculated as [units of corresponding state variable]$^2/[rad/sec]$. For example, the units of the average noise strength corresponding to velocity, $\mathbf{Q}(1,1)$ in Table 3.5, are $[ft/sec]^2/[rad/sec] = ft^2/rad \cdot sec$.

The noise injection matrix used to introduce the dynamics noise into the continuous-time model is presented by Eide [8], and is given in Equation (3.14) in terms of the primed dimensional stability derivatives from Section 3.4.1. Note that the fourth and fifth columns of **G** are interchanged with respect to the order of the state vector given in Table 3.1. This discrepancy results from a different order of these same variables in the **Q** matrix, as given in Table 3.5.

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ X'_u & X'_\alpha & X'_q & 0 & 0 & 0 \\ Z'_u & Z'_\alpha & Z'_q & 0 & 0 & 0 \\ M'_u & M'_\alpha & M'_q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Y'_p & Y'_\beta & Y'_r \\ 0 & 0 & 0 & L'_p & L'_\beta & L'_r \\ 0 & 0 & 0 & N'_p & N'_\beta & N'_r \end{bmatrix} \quad (3.14)$$

3.4.4 Actuator Models.

The linear model supplied by the SRF VISTA simulation does not incorporate any actuator models, naively assuming an instantaneous response. A more realistic response is achieved by using the fourth order servo-actuator models given on the Block 40 functional diagram [11]:

$$\frac{\delta_{act}}{\delta_{cmd}} = \frac{(20.2)(144.8)(71.4)^2}{(s+20.2)(s+144.8)(s^2+2(0.736)(71.4)s+71.4^2)} \tag{3.15}$$

Note that the same fourth order model is given for all five of the control surfaces. For the design model, the actuators are modeled as first order lags to reduce computational loading from that of higher order models; this does not degrade model adequacy in any appreciable manner. The break point selected for the actuators is $\omega = 14$, which has the response empirically determined to match the SRF VISTA actuator simulation best [8]. Using a generic label, the actuator states then have the form:

$$\dot{\delta}_{act} = -14\delta_{act} + 14\delta_{cmd} \tag{3.16}$$

These states are then augmented to the state-space model, yielding

$$\dot{\mathbf{x}}_{aug}(t) = \begin{bmatrix} \mathbf{A} & \mathbf{B}_{mod} \\ \mathbf{0} & -14\cdot\mathbf{I} \end{bmatrix} \mathbf{x}_{aug}(t) + \begin{bmatrix} \mathbf{0} \\ 14\cdot\mathbf{I} \end{bmatrix} \mathbf{u}_{aug}(t) + \begin{bmatrix} \mathbf{G} \\ \mathbf{0} \end{bmatrix} \mathbf{w}(t) \tag{3.17}$$

$$\mathbf{z}(t_i) = \begin{bmatrix} \mathbf{H} & \mathbf{D}_z \end{bmatrix} \mathbf{x}_{aug}(t_i) + \mathbf{v}(t_i) \tag{3.18}$$

$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{C} & \mathbf{D}_y \end{bmatrix} \mathbf{x}_{aug}(t_i) \tag{3.19}$$

where the augmented state and input vectors are shown in Tables 3.6 and 3.7. Note that the leftmost column of these two tables specifies the component index of the named vector, and that the first eight components of the augmented state vector are identical to the eight states of the original state vector, given in Table 3.1.

Table 3.6 Augmented State Vector

$\mathbf{x}_{aug}$	Augmented State Variables		Units
9	δ_{ls}	Left stabilator position	rad
10	δ_{rs}	Right stabilator position	rad
11	δ_{lf}	Left flaperon position	rad
12	δ_{rf}	Right flaperon position	rad
13	δ_{rud}	Rudder position	rad

Table 3.7 Augmented Input Vector

$\mathbf{u}_{aug}$	Augmented Input Variables		Units
1	δ_{ls-c}	Left stabilator command	rad
2	δ_{rs-c}	Right stabilator command	rad
3	δ_{lf-c}	Left flaperon command	rad
4	δ_{rf-c}	Right flaperon command	rad
5	δ_{rud-c}	Rudder command	rad

3.4.5 Equivalent Discrete-Time Model. Because the MMAC will be implemented on a digital computer, the last step in generating the design model is to convert Equations (3.17) - (3.19) to an equivalent discrete-time representation [23, 43] based on a sampling frequency of 64 Hz. In truth, the measurement model is already in a discrete-time form, and the output model is carried over with no change. The appropriate terms for the dynamics model are computed for the LTI system as [43]:

$$\mathbf{\Phi} = e^{\mathbf{A}_{aug}\Delta T} \tag{3.20}$$

$$\mathbf{B}_d = \left(\int_0^{\Delta T} e^{\mathbf{A}_{aug}\tau} d\tau\right) \mathbf{B}_{aug} \tag{3.21}$$

where ΔT is the sample period. The appropriate discrete-time white noise, $\mathbf{w}_d$, has the statistics [23]:

$$E\{\mathbf{w}_d(t_i)\} = \mathbf{0} \tag{3.22}$$

$$E\{\mathbf{w}_d(t_i)\mathbf{w}_d^T(t_i)\} = \mathbf{Q}_d = \int_0^{\Delta T} e^{\mathbf{A}_{aug}\tau}\mathbf{GQG}^T e^{\mathbf{A}_{aug}^T\tau} d\tau \tag{3.23}$$

$$E\{\mathbf{w}_d(t_i)\mathbf{w}_d^T(t_j)\} = \mathbf{0}, \quad t_i \neq t_j \tag{3.24}$$

The resulting system description is summarized as:

$$\begin{aligned}
\mathbf{x}_{aug}(t_{i+1}) &= \mathbf{\Phi}\mathbf{x}_{aug}(t_i) + \mathbf{B}_d\mathbf{u}_{aug}(t_i) + \mathbf{w}_d(t_i) \\
\mathbf{z}(t_i) &= \mathbf{H}_{aug}\mathbf{x}_{aug}(t_i) + \mathbf{v}(t_i) \\
\mathbf{y}(t_i) &= \mathbf{C}_{aug}\mathbf{x}_{aug}(t_i)
\end{aligned} \tag{3.25}$$

3.5 Failure Models

Both actuator and sensor failure models are incorporated into both the design and the truth models. In the linear design model, failures are modeled by multiplying the appropriate column of the input matrix (for actuator failures) or row of the measurement matrix (for sensor failures) by a scalar, ϵ, where $0 \leq \epsilon \leq 1$, with the '0' representing a complete failure and '1' representing a fully functional component. Note that a completely failed actuator can be regarded as either an actuator failure to free stream, in which the surface simply floats in the relative wind, or as a complete loss of the surface, possibly due to damage. Also, sensor failures still admit the sensor noise into the measurement vector. Incorporating the failure modes results into the continuous-time dynamics and sampled-data measurement equations yields:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{aug}\mathbf{x}(t_i) + \mathbf{B}_{aug}\mathbf{F}_{ai}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t) \tag{3.26}$$

$$\mathbf{z}(t_i) = \mathbf{F}_{sj}\mathbf{H}_{aug}\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{3.27}$$

where $\mathbf{F}_{ai}$ and $\mathbf{F}_{sj}$ are identity matrices of appropriate size except that $\mathbf{F}_{ai}(i,i) = \epsilon$ if the i^{th} actuator has failed and $\mathbf{F}_{sj}(j,j) = \epsilon$ if the j^{th} sensor has failed. The failures are incorporated before the system is converted to the discrete time to reflect the fact that the truth model, where the failures will actually be inserted, uses continuous-time dynamics.

Implementation of the sensor failures in the nonlinear truth model can be accomplished in much the same manner as in the design model; the noise-free sensor reading is multiplied by ϵ, and the sensor noise is admitted into the measurement unchanged. Modeling the actuator failures, however, is more difficult. Ideally, the portion of the aerodynamic equations which calculate the failed surface's contribution would be scaled by ϵ. Unfortunately, due to time restrictions, a detailed analysis of the implication of surface failure on the aerodynamic computations is not possible, so actuator failures are instead modeled by commanding the appropriate surface to its trim position. The trim position is used for two reasons. First, it approximates the "failure to free stream" position that was assumed with the linear design model. Secondly, although the (linear perturbation) design model's actuator failures effectively zero out the commanded (perturbation) position (for a complete failure), the nominal position remains unaffected, so the true failed position on the design model (i. e., zeroed perturbation variable) does indeed equal the trim position (for total versus perturbation variables), just like the truth model's failure position.

In reality, more complex failure modes may be experienced, such as biased sensors, stuck actuators, or a deterioration in the actuator's time response. Battle damage may even contribute to the loss of part or all of a control surface. These failures are more accurately modeled by changes in the stability derivatives, effecting the **A** matrix in addition to the **B** and **H** matrices. However, these simplified failure modes are used for two reasons. First they are easy to implement and they accurately model a fair range of possible failures. Second, through the use of Bayesian blending, the MMAC may still provide suitable control of the more complex failure modes, and an extension of this research may be to add the additional failure modes to the truth model to verify this ability.

Due to time constraints, this research will focus only on single, complete failures, leaving double failures and partial failures to future consideration. A side effect is that the bank swapping associated with a hierarchical structure will no longer be implemented. Technically, the hierarchical structure still exists, but with only a depth of one, and therefore only one bank.

3.6 LQG Synthesis

The general control problem, pictured in Figure 3.6, is representative of all elemental filter/controller pairs in the MMAC. The output of the plant, in this case the VISTA F-16, is required to track a reference input in the presence of both modeled and unmodeled disturbances. These disturbances include wind gusts and the effects of higher order terms neglected in linearizing the design model. Recognizing that full-state feedback is not available on the VISTA F-16, and that the available measurements are corrupted by noise due to imperfect sensors, Kalman filters, incorporated within the block labeled "Controller" in Figure 3.6, provide optimal state estimates which drive the deterministic controllers.

One controller is needed for the fully functional aircraft, with an additional controller required for each actuator failure case. Note that the sensor failures do not necessitate a unique controller since the estimates from the Kalman filters already compensate for missing sensor information. Thus the controllers in Figure 2.3 for failed-sensor hypotheses will be the same as the controller for the fully functional aircraft, namely the original Block 40 flight control system already implemented in the SRF VISTA simulation.

An LQG controller for each failed-actuator hypothesis is selected based on several factors. First, it yields "readily synthesized, efficiently implemented, feedback control laws" [24]. Second, it has been successfully implemented with excellent results in previous MMAC designs [20, 29, 31]. Also, as a modern technique the LQG design method is well suited for the multi-variable aircraft

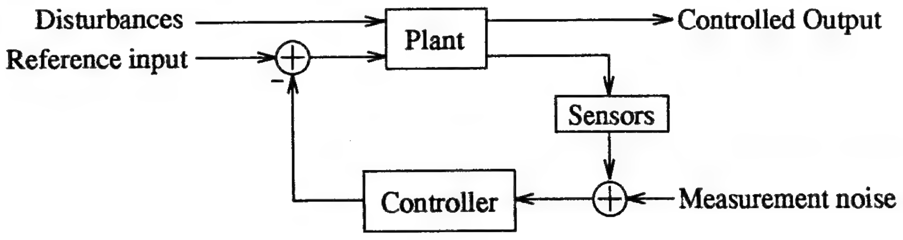


Figure 3.6 General Control Problem

control problem. Finally, through use of the certainty equivalence property [24], the LQG controller can be designed assuming full state feedback is available, with an independently designed Kalman filter providing state estimates in the absence of full state feedback. Certainty equivalence is especially applicable in this research since full state feedback is not available and the Kalman filters have already been designed for the MMAE-based controller of the previous research.

The three assumptions of the LQG controller are easily accommodated. The design model is a linear, discrete-time model. The quadratic cost function is well suited to aerospace problems since it attempts to minimize deviations in the state and control vectors, thereby helping to validate the linear perturbation models. Finally, many aircraft disturbances, such as sensor noise and wind gusts, have successfully been modeled in the past by Gaussian noise, and the central limit theorem provides justification for using Gaussian noise models as being either the best models or very close approximations to best models for physically observed noises [23].

In the following sections the LQ full-state feedback controller will be reviewed as the second essential building block for the LQG controller, to be augmented to the already designed Kalman filter. As Type I properties are desired, PI compensation is introduced to the controller. Implicit and explicit model following techniques are then described which could be added, if necessary, to enhance tracking performance or robustness.

3.6.1 LQ Regulator. The objective of the LQ regulator [24] is to determine the control function, $\mathbf{u}^*$ (for consistency with the development of the design model, the augmented vectors, $\mathbf{x}_{aug}$ and $\mathbf{u}_{aug}$, from Equation (3.25) should be used here and throughout the remainder of this chapter, but for notational convenience, the subscripts are dropped), which minimizes the quadratic cost function

$$J = \frac{1}{2}\sum_{i=0}^{N}\begin{bmatrix}\mathbf{x}(t_i)\\ \mathbf{u}(t_i)\end{bmatrix}^T\begin{bmatrix}\mathbf{X}(t_i) & \mathbf{S}(t_i)\\ \mathbf{S}^T(t_i) & \mathbf{U}(t_i)\end{bmatrix}\begin{bmatrix}\mathbf{x}(t_i)\\ \mathbf{u}(t_i)\end{bmatrix} + \frac{1}{2}\mathbf{x}^{\mathbf{T}}(t_{N+1})\mathbf{X}_f(t_{N+1})\mathbf{x}(t_{N+1}) \qquad (3.28)$$

where $\mathbf{X}$ penalizes deviations of the states from zero, $\mathbf{U}$ penalizes deviations of the control signal from zero, and $\mathbf{X}_f$ penalizes deviations of the states from zero at the final time, t_{N+1}. Note that the control signal is not weighted at the final time since there is no physical reason to expend control authority after the problem has already finished (by contrast, a final state, such as position, may be critical.) $\mathbf{S}$ represents cross-weighting between the states and inputs which arises due to weighting on output variables dependent on the derivatives of states or due to the discretization of a cost function originally posed in continuous-time as an integral instead of a summation [24]. If the time horizon, as in this case, is stretched to infinity, the final state is essentially never reached and can be disregarded. Therefore, the second term in Equation (3.28) is removed when the intent is to synthesize such an infinite-horizon controller.

If one wishes to regulate an output vector, $\mathbf{y}$, instead, the cost function can be written (neglecting the penalty on the final state) as

$$J = \frac{1}{2}\sum_{i=0}^{N}\left[\mathbf{y}^T(t_i)\mathbf{Y}(t_i)\mathbf{y}(t_i) + \mathbf{u}^T(t_i)\mathbf{U}_0(t_i)\mathbf{u}(t_i)\right] \tag{3.29}$$

This cost function can be recast into the form of

$$J = \frac{1}{2}\sum_{i=0}^{N}\begin{bmatrix}\mathbf{x}(t_i)\\ \mathbf{u}(t_i)\end{bmatrix}^T\begin{bmatrix}\mathbf{X}(t_i) & \mathbf{S}(t_i)\\ \mathbf{S}^T(t_i) & \mathbf{U}(t_i)\end{bmatrix}\begin{bmatrix}\mathbf{x}(t_i)\\ \mathbf{u}(t_i)\end{bmatrix} \tag{3.30}$$

if the following substitutions are made:

$$\mathbf{X}(t_i) = \mathbf{C}^T\mathbf{Y}(t_i)\mathbf{C} \tag{3.31}$$

$$\mathbf{S}(\mathbf{t_i}) = \mathbf{C}^T\mathbf{Y}(t_i)\mathbf{D} \tag{3.32}$$

$$\mathbf{U}(t_i) = \mathbf{U}_0(t_i) + \mathbf{D}^T\mathbf{Y}(t_i)\mathbf{D} \tag{3.33}$$

From this definition of the cost function, the LQ control laws can be synthesized. The reader is assumed to be familiar with the basic LQ regulator, so these equations are presented below without proof (Maybeck presents a more detailed derivation in *Stochastic Models, Estimation, and Control Volume 3* [24]). Assuming **X**, **S**, and **U** to be constants, and ignoring initial and terminal transients (by letting N go to infinity in Equation (3.30)), the steady-state constant-gain control law is

$$\mathbf{u}(t_i) = -\mathbf{G}_c^* \mathbf{x}(t_i) \tag{3.34}$$

where

$$\mathbf{G}_c^* = [\mathbf{U} + \mathbf{B}_d^T \mathbf{K}_c \mathbf{B}_d]^{-1} [\mathbf{B}_d^T \mathbf{K}_c \mathbf{\Phi} + \mathbf{S}] \tag{3.35}$$

and

$$\mathbf{K}_c = [\mathbf{\Phi} - \mathbf{B}_d \mathbf{G}_c^*]^T \mathbf{K}_c [\mathbf{\Phi} - \mathbf{B}_d \mathbf{G}_c^*] + \mathbf{G}_c^{*T} \mathbf{U} \mathbf{G}_c^* + \mathbf{X} - \mathbf{S}\mathbf{G}_c^* - \mathbf{G}_c^{*T} \mathbf{S}^T \tag{3.36}$$

Note that the the state vector, $\mathbf{x}(t_i)$, in Equation (3.34) gets replaced by the estimate, $\hat{\mathbf{x}}(t_i^+)$, using certainty equivalence. Optionally, the sub-optimal controller, using $\hat{\mathbf{x}}(t_i^-)$, could be used to keep computational delay time to a minimum.

3.6.2 PI Control. The desire to achieve Type I performance characteristics, i. e. to track a constant reference input with zero steady-state mean error despite unmodeled constant disturbances, motivates the need for proportional plus integral (PI) control. Consider the closed loop system shown in Figure 3.7, where the error vector is defined as $\mathrm{e}(t_i) = \mathbf{r} - \mathbf{y}(t_i)$[1]. For a stable system in steady state, the error, and thus the control input, approaches zero. For a linear system, a zero input vector will drive the output to zero as well. While this type of response is appropriate for a regulator, the system as is cannot track a reference input while maintaining zero error. For tracking, the controller must be able to deliver a nonzero steady state control when

[1]Note the omission of a time dependence on the reference signal, **r**. The implicit assumption is that **r** is piecewise constant, not overly restrictive since the sampling rate for the VISTA F-16 is a fast 64 Hz.

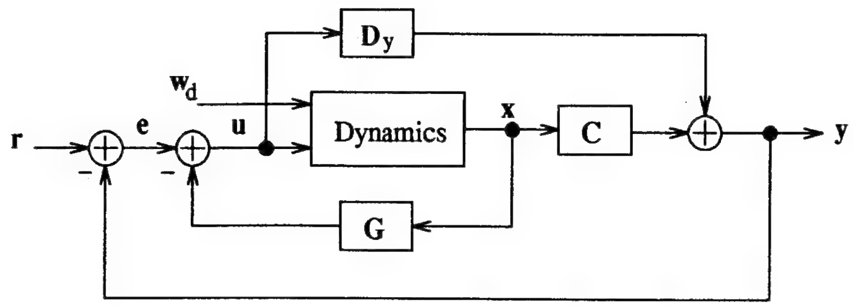


Figure 3.7 Potential Controller Design Based on Full-State LQ Regulator

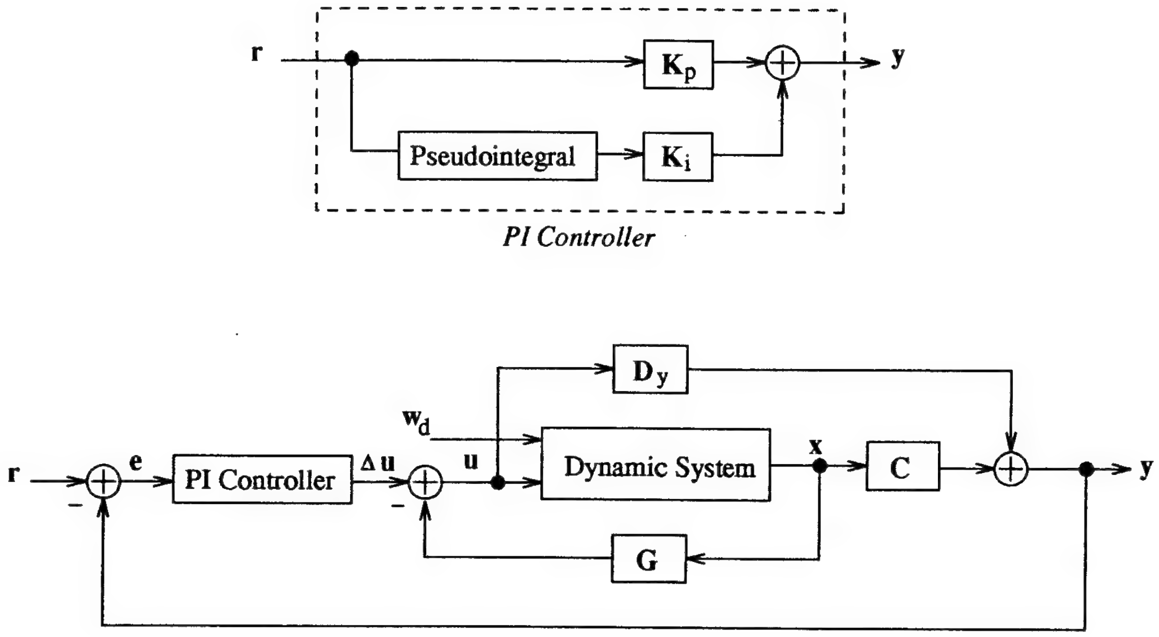


Figure 3.8 Closed Loop System with Position Form PI Control Law

its own input is zero [24]. This necessary characteristic can be realized by augmenting integrators (actually pseudointegrators, or summations, for the discrete time problem) such that the integral channel maintains a nonzero value even after the proportional channel (corresponding to the error) has gone to zero. The LQG/PI controller gains ($\mathbf{K}_p$, $\mathbf{K}_i$, and $\mathbf{G}$) of Figure 3.8, are then computed by applying the LQ regulator synthesis to the augmented system. These pseudointegrators may be augmented either before or after the original plant model to define the augmented system for use in the LQ regulator synthesis. In the former case the augmented states represent the derivative of the control input, while in the latter implementation they describe the integral of the regulation error. Because of physical limitations in the aircraft actuators, the latter method is used here so that a quadratic penalty can be applied to the actuator rates. For reference, the deflection and rate limits incorporated in the SRF VISTA simulation are given in Table 3.8. Note that when, as in this research, the true plant states, $\mathbf{x}$, are not really accessible, noise-corrupted measurements, $\mathbf{z}$, are extracted from the system and used as inputs to a Kalman filter, which generates $\hat{\mathbf{x}}$ to use in place of $\mathbf{x}$ in Figure 3.8.

Table 3.8 Actuator Deflection and Rate Limits

Actuator	Upper Limit (deg)	Lower Limit (deg)	Rate Limit (deg/sec)
Left Stabilator	23	-19	60
Right stabilator	23	-19	60
Left flaperon	21.5	-21.5	62
Right flaperon	21.5	-21.5	62
Rudder	30	-30	120

If the pseudorate is defined to be the difference between the control signal at two consecutive sampling instants, $\mathbf{\Delta u}(t_i) = \mathbf{u}(t_{i+1}) - \mathbf{u}(t_1)$, then the original plant states can be augmented with the pseudorate states, resulting in:

$$\begin{bmatrix} \delta\mathbf{x}(t_{i+1}) \\ \delta\mathbf{u}(t_{i+1}) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi} & \mathbf{B}_d \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}(t_i) \\ \delta\mathbf{u}(t_i) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{\Delta u}(t_i) \qquad (3.37)$$

$$\mathbf{y}_c(t_i) = \begin{bmatrix} \mathbf{C}_{aug} & \mathbf{0} \end{bmatrix} \delta\mathbf{x}(t_i) \qquad (3.38)$$

Careful inspection reveals that the PI controller synthesis has now been recast as a regulator with a new cost function given by

$$J = \frac{1}{2}\sum_{i=0}^{N} \begin{bmatrix} \delta\mathbf{x}(t_i) \\ \delta\mathbf{u}(t_i) \end{bmatrix}^T \begin{bmatrix} \mathbf{X} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{U} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}(t_i) \\ \delta\mathbf{u}(t_i) \end{bmatrix} + \frac{1}{2}\mathbf{\Delta u}(t_i)^T \mathbf{U}_R \mathbf{\Delta u}(t_i) \qquad (3.39)$$

where the new quadratic weighting matrix, $\mathbf{U}_R$, is used to penalize deviations of the pseudorates from zero. The steady-state, constant-gain control law (again letting $N \rightarrow \infty$) is calculated using Equations (3.34)-(3.36). By partitioning the augmented state vector as $[\ \delta\mathbf{x}^T \quad \delta\mathbf{u}^T\ ]^T$, the feedback gain, $\mathbf{G}_c^*$ can be partitioned as $[\ \mathbf{G}_{c1}^* \quad \mathbf{G}_{c2}^*\ ]$, allowing the control law to be written in an incremental form as

$$\delta\mathbf{u}^*(t_{i+1}) \quad = \quad \delta\mathbf{u}^*(t_i) - \mathbf{G}_{c1}^*\delta\mathbf{x}(t_i) - \mathbf{G}_{c2}^*\delta\mathbf{u}^*(t_i) \tag{3.40}$$

Despite the addition of pseudointegrators to the model, some manipulation of the control law is still required to attain Type I properties. The desired form of the control law is

$$\mathbf{u}(t_i) = \mathbf{u}(t_{i-1}) - \mathbf{K}_x\left[\mathbf{x}(t_i) - \mathbf{x}(t_{i-1})\right] + \mathbf{K}_\xi\left[\mathbf{r} - \mathbf{y}(t_{i-1})\right] \tag{3.41}$$

where the last feedback term, $\mathbf{r} - \mathbf{y}(t_{i-1})$, is the regulation error intended to be driven to zero. The following extract from *Stochastic Models, Estimation, and Control Volume 3* [24] shows the steps necessary to massage the control law into this form.

Written in terms of perturbation states, and substituting in for $\delta\mathbf{y}(t_i)$ using Equation (3.25) (expressed in terms of perturbations), Equation (3.41) becomes

$$\delta\mathbf{u}(t_{i+1}) \quad = \quad \delta\mathbf{u}(t_i) - \mathbf{K}_x\left[\delta\mathbf{x}(t_{i+1}) - \delta\mathbf{x}(t_i)\right] + \mathbf{K}_\xi\left[-\delta\mathbf{y}(t_i)\right] \tag{3.42}$$

$$= \quad \delta\mathbf{u}(t_i) - \mathbf{K}_x\left[\delta\mathbf{x}(t_{i+1}) - \delta\mathbf{x}(t_i)\right] - \mathbf{K}_\xi\left[\mathbf{C}_{aug}\delta\mathbf{x}(t_i)\right] \tag{3.43}$$

Manipulating Equation (3.25) to get $\delta\mathbf{x}(t_{i+1}) - \delta\mathbf{x}(t_i) = \left[\mathbf{\Phi} - \mathbf{I}\right]\delta\mathbf{x}(t_i) + \mathbf{B}_d\delta\mathbf{u}(t_i)$, Equation (3.43) can be rewritten as

$$\delta\mathbf{u}(t_{i+1}) = \delta\mathbf{u}(t_i) - \begin{bmatrix} \mathbf{K}_x & \mathbf{K}_\xi \end{bmatrix} \begin{bmatrix} \mathbf{\Phi} - \mathbf{I} & \mathbf{B}_d \\ \mathbf{C}_{aug} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}(t_i) \\ \delta\mathbf{u}(t_i) \end{bmatrix} \tag{3.44}$$

Comparing this result to that obtained from the LQ regulator synthesis, Equation (3.40), it is seen that the two forms are equivalent if $\mathbf{K}_x$ and $\mathbf{K}_\xi$ are chosen to satisfy

$$\begin{bmatrix} \mathbf{K}_x & \mathbf{K}_\xi \end{bmatrix} \begin{bmatrix} \mathbf{\Phi} - \mathbf{I} & \mathbf{B}_d \\ \mathbf{C}_{aug} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{G}^*_{c1} & \mathbf{G}^*_{c2} \end{bmatrix} \tag{3.45}$$

or

$$\begin{aligned} \begin{bmatrix} \mathbf{K}_x & \mathbf{K}_\xi \end{bmatrix} &= \begin{bmatrix} \mathbf{G}^*_{c1} & \mathbf{G}^*_{c2} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi} - \mathbf{I} & \mathbf{B}_d \\ \mathbf{C}_{aug} & \mathbf{0} \end{bmatrix}^{-1} && (3.46) \\ &= \begin{bmatrix} \mathbf{G}^*_{c1} & \mathbf{G}^*_{c2} \end{bmatrix} \begin{bmatrix} \mathbf{\Pi}_{11} & \mathbf{\Pi}_{12} \\ \mathbf{\Pi}_{21} & \mathbf{\Pi}_{22} \end{bmatrix} && (3.47) \end{aligned}$$

which yields the final result of

$$\begin{aligned} \mathbf{K}_x &= \mathbf{G}^*_{c1}\mathbf{\Pi}_{11} + \mathbf{G}^*_{c2}\mathbf{\Pi}_{21} && (3.48) \\ \mathbf{K}_\xi &= \mathbf{G}^*_{c1}\mathbf{\Pi}_{12} + \mathbf{G}^*_{c2}\mathbf{\Pi}_{22} && (3.49) \end{aligned}$$

3.6.3 Implicit and Explicit Model Following. Two model following techniques are available to enhance the performance and/or robustness of the LQG controller. They may also be used to attempt to match the structure of the Block 40 flight control system more closely, as described in Section 2.3.5. The first technique is implicit model following [4] which penalizes deviations from a desired trajectory by modifying the existing feedback gains via the quadratic weighting terms. Performance is not necessarily greatly affected, nor is the controller dimensionality affected at all, but because the feedback is modified, robustness can be enhanced. Note that the traditional technique of robustness enhancement for an LQG controller, LTR tuning [24], is not applicable for the MMAC since the increased pseudonoise strength in the filter design dynamics model incapacitates the adaptation process [35].

The desired output is modeled by

$$\mathbf{y}_i(t_{i+1}) = \mathbf{\Phi}_i \mathbf{y}_i(t_i) \tag{3.50}$$

System robustness is maximized if the eigenvectors are orthogonal [13, 32]. The simplest model satisfying this condition is a diagonal state transition matrix (block diagonal if complex eigenvalues are used):

$$\mathbf{\Phi}_i = \begin{bmatrix} \mathbf{\Phi}_{i1} & & & \\ & \mathbf{\Phi}_{i2} & & \\ & & \ddots & \\ & & & \mathbf{\Phi}_{in} \end{bmatrix} \tag{3.51}$$

Equation (3.50) can be rearranged and incorporated into the cost function as

$$J = \frac{1}{2}\sum_{0}^{N} \left[(\mathbf{y}(t_{i+1}) - \mathbf{\Phi}_i \mathbf{y}(t_i))^T \mathbf{Y}_I (\mathbf{y}(t_{i+1}) - \mathbf{\Phi}_i \mathbf{y}(t_i)) + \mathbf{u}(t_i)^T \mathbf{U}_{I0} \mathbf{u}(t_i) \right] \tag{3.52}$$

Substituting with the discrete-time model shown in Equation (3.25), and removing the time arguments for convenience, the cost function can also be expressed as:

$$J = \frac{1}{2}\sum_{i=0}^{N} \begin{bmatrix} \mathbf{x}(t_i) \\ \mathbf{u}(t_i) \end{bmatrix}^T \begin{bmatrix} \mathbf{X}_I(t_i) & \mathbf{S}_I(t_i) \\ \mathbf{S}_I^T(t_i) & \mathbf{U}_{I0}(t_i) \end{bmatrix} \begin{bmatrix} \mathbf{x}(t_i) \\ \mathbf{u}(t_i) \end{bmatrix} \tag{3.53}$$

where

$$\mathbf{X}_I(t_i) = (\mathbf{C}\mathbf{\Phi} - \mathbf{\Phi}_i \mathbf{C})^T \mathbf{Y}_I(t_i) (\mathbf{C}\mathbf{\Phi} - \mathbf{\Phi}_i \mathbf{C}) \tag{3.54}$$

$$\mathbf{U}_I(t_i) = \mathbf{U}_{I0}(t_i) + \mathbf{B}_d^T \mathbf{C}^T \mathbf{Y}_I(t_i) \mathbf{C} \mathbf{B}_d \tag{3.55}$$

$$\mathbf{S}_I(t_i) = (\mathbf{C}\mathbf{\Phi} - \mathbf{\Phi}_i \mathbf{C})^T \mathbf{Y}_I(t_i) \mathbf{C} \mathbf{B}_d \tag{3.56}$$

Conveniently, this form is now that of the standard LQ Regulator, and the control law can be evaluated using Equations (3.34-3.36), again letting $N \to \infty$ to achieve a constant-gain steady state design.

The second method is explicit model following through use of a command generator tracker (CGT), shown in Figure 3.9, where the controlled system can represent either a simple LQ regulator or a regulator incorporating PI control for tracking. The CGT forces the plant to follow a desired output model, $\mathbf{y}_m$, generated by

$$\begin{aligned} \mathbf{x}_m(t_{i+1}) &= \mathbf{\Phi}_m \mathbf{x}_m(t_i) + \mathbf{B}_m \mathbf{u}_m(t_i) \\ \mathbf{y}_m(t_i) &= \mathbf{C}_m \mathbf{x}_m(t_i) + \mathbf{D}_m \mathbf{u}_m(t_i) \end{aligned} \tag{3.57}$$

From the block diagram, it is apparent that the resulting CGT/PI controller is a feed-forward design, and as such it does not provide robustness enhancement, a primary concern of this research. Also, the states of the explicit model are appended to the original plant, increasing the dimensionality. To avoid the additional computational burden, explicit model following will not be used in this research unless the required performance characteristics cannot be achieved with PI control alone.

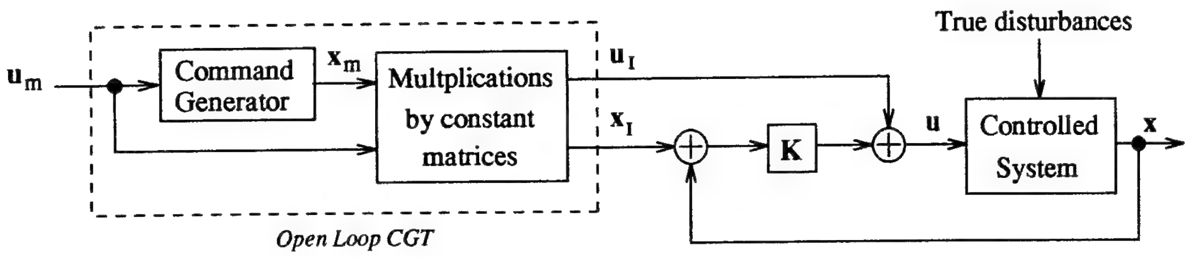


Figure 3.9 Closed Loop Command Generator Tracker

3.7 LQG Design Procedure

Before the MMAC can be assembled, a few steps must be taken. First, the SRF VISTA software must be modified to allow the inclusion of the MMAC. In doing so, the existing MMAE structure will be exploited to the fullest extent. As a first step, the MMAE function call must be relocated from its current location at the end of the simulation loop to a point before the controller function call. Doing so will remove a half sample period delay (the SRF VISTA updates the airframe dynamics at 128 Hz - twice the frequency of the flight control system), necessitating the need to retune the Kalman filters.

Next, a PI (or CGT/PI if required) controller must be synthesized which can reproduce the performance of the existing Block 40 flight control system as closely as possible. The steps used in the synthesis are summarized as:

1. Generate the plant and input matrices: $\mathbf{A}$ and $\mathbf{B}$
2. Specify the measurement and output models: $\mathbf{H}$, $\mathbf{D}_z$, $\mathbf{C}$, and $\mathbf{D}_y$
3. Augment the actuator models
4. Convert augmented system to discrete time
5. Specify the initial tracking weighting matrices: $\mathbf{Y}$ and $\mathbf{U}$
6. Calculate the LQ regulator weighting matrices: $\mathbf{X}$, $\mathbf{S}$, and $\mathbf{U}$
7. Specify the weighting matrix for the pseudorates: $\mathbf{U}_R$
8. Specify the implicit and explicit models (if used)
9. Generate the feedback gain, $\mathbf{G}_c^*$
10. Evaluate the closed loop system response, iteratively tuning the weighting matrices as needed

Based on this initial controller, additional controllers will be designed for each failure hypothesis involving at least one actuator failure. As described in Section 2.3.4, the actual Block 40

FCS will be used for the controller corresponding to the fully functional hypothesis and hypotheses involving only sensor failures.

Several MATLAB script files have been written to assist in tuning and evaluating the Kalman filters and LQG controllers: *filtergen.m*, *lqggen.m*, *probplot.m* and *stateplot.m*. Additional software has been written to facilitate the interaction with the SRF VISTA during the iterative tuning processes: *monte*, *singlerun*, and *tune*.

3.8 Control Redistribution Synthesis

As an alternative to replacing or modifying the existing, tested, and well-performing Block 40 flight control system, the method of control redistribution leaves the Block 40 intact. As depicted in Figure 3.10, MMAE-based control redistribution uses an MMAE as a front-end to provide parameter and state estimation, and control redistribution as a back-end to process the actuator commands sent from the Block 40. The basic premise of control redistribution is that sufficient redundancy is inherent in the existing aircraft control surfaces so that commands intended for a failed control surface can be redistributed to the remaining functional control surfaces with virtually no noticeable change in performance. Although control redistribution cannot completely compensate for the loss of an actuator (the total amount of control authority available to any flight control system is decreased an amount commensurate with the control authority possessed by the

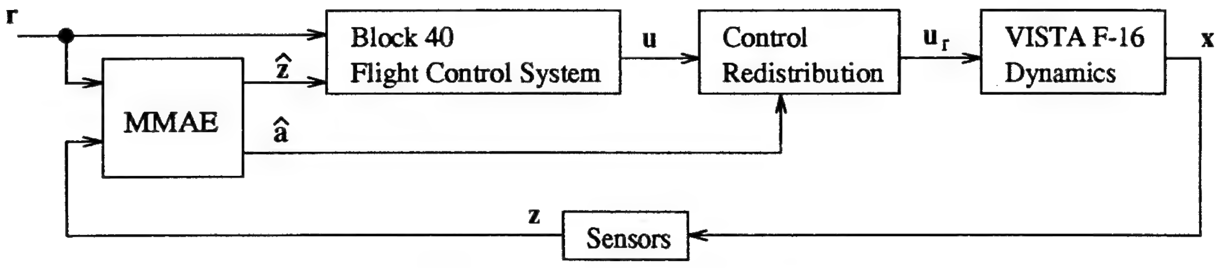


Figure 3.10 Control Redistribution

individual actuator before it failed), for small to moderate commands, the drop in performance will be shown in Chapter 4 to be negligible.

Mathematically, the desired redistribution relationship (redistributed control applied to the failed system is equivalent to control applied to the fully functional system) can be expressed as

$$\mathbf{B}_{fail}\mathbf{u}_r \approx \mathbf{B}\mathbf{u} \tag{3.58}$$

where $\mathbf{B}_{fail}$ represents the input matrix corresponding to a failed actuator condition and $\mathbf{u}_r$ represents the redistributed control signals. The need for an approximate relationship (versus a true equality) will become apparent later, though physically one might expect that the use of redistributed control will not have exactly the same contribution to the dynamics as using the original control (unless the actuators in question truly are redundant components). Recall from Section 3.5 that the failed input matrix has one or more columns scaled by a constant, $0 \leq \epsilon \leq 1$. As in the MMAE, the redistribution matrix will be constructed using completely failed assumptions, $\epsilon = 0$. Using the failure model developed in Section 3.5, the failed input matrix is expressed as:

$$\mathbf{B}_{fail} = \mathbf{B}\mathbf{F}_{ai} \tag{3.59}$$

Assuming a linear transformation for the redistribution matrix, the redistribution control input becomes:

$$\mathbf{u}_r = \mathbf{D}_{ai}\mathbf{u} \tag{3.60}$$

where the notation $\mathbf{D}_{ai}$ represents the redistribution matrix corresponding to complete failure of the i^{th} actuator. Note that a redistribution matrix corresponding to a double actuator failure condition, $\mathbf{D}_{i,j}$ can also be synthesized by using the combined actuator failure matrix, $\mathbf{F}_{ai,j} = \mathbf{F}_{ai}\mathbf{F}_{aj}$, in

Equation (3.59). Substituting Equations (3.59) and (3.60) into Equation (3.58) yields:

$$\mathbf{B}\mathbf{F}_{ai}\mathbf{D}_{ai}\mathbf{u} \approx \mathbf{B}\mathbf{u} \tag{3.61}$$

Because the solution must be true for a general control input, **u** is removed from both sides of Equation (3.61), leaving:

$$\mathbf{B}\mathbf{F}_{ai}\mathbf{D}_{ai} \approx \mathbf{B} \tag{3.62}$$

The temptation to remove **B** from both sides must be avoided since no solution for **D** exists for the expression $\mathbf{F}_{ai}\mathbf{D}_{ai} = \mathbf{I}$ when a complete failure is hypothesized. The problem is that, when $\mathbf{F}_{ai}$ is rank deficient (recall that failed actuators are indicated by columns of zeroes), a pseudoinverse is needed to find a solution, and, whereas for a true inverse the equality, $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$, exists, this relationship does not hold for a pseudoinverse. The solution for the redistribution matrix is then:

$$\mathbf{D}_{ai} \approx (\mathbf{B}\mathbf{F}_{ai})^{+}\mathbf{B} \tag{3.63}$$

where the pseudoinverse is denoted by the superscript +. The need for the approximation back in Equation (3.58) is now clear: While the pseudoinverse will give a true inverse where possible, in the direction corresponding to the deficient column, the pseudoinverse gives only an approximate solution. The pseudoinverse does, however, give the best solution in the least squares sense, i. e. the error, $\| \mathbf{B}\mathbf{F}_{ai}\mathbf{D}_{ai} - \mathbf{B} \|$ is minimized [47].

At this point, an analysis of the redistribution matrix, $\mathbf{D}_{ai}$, is warranted. $\mathbf{D}_{ai}$ has the form of an identity matrix except that the i^{th} column has been replaced by a column which redistributes the i^{th} control input to the remaining inputs. This property can be shown by simple manipulation of Equation (3.63):

$$\mathbf{D}_{ai} \approx (\mathbf{B}\mathbf{F}_{ai})^{+}\mathbf{B} \tag{3.64}$$

$$\approx \quad (\mathbf{BF}_{ai})^{+}\mathbf{B} - \mathbf{F}_{ai} + \mathbf{F}_{ai} \tag{3.65}$$

$$\approx \quad (\mathbf{BF}_{ai})^{+}\mathbf{B} - (\mathbf{BF}_{ai})^{+}(\mathbf{BF}_{ai}) + \mathbf{F}_{ai} \tag{3.66}$$

$$\approx \quad (\mathbf{BF}_{ai})^{+}\mathbf{B}(\mathbf{I} - \mathbf{F}_{ai}) + \mathbf{F}_{ai} \tag{3.67}$$

The second term of Equation (3.67) creates a diagonal matrix with ones in the diagonal entry of every column corresponding to a functional actuator. The first term, on the other hand, pulls off the failed columns of $(\mathbf{BF}_{ai})^{+}\mathbf{B}$ and adds them to the this diagonal matrix to create $\mathbf{D}_{ai}$. As an example, consider the redistribution matrix corresponding to an assumed failed left stabilator, given for this research as:

$$\mathbf{D}_{a1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0.9060 & 0 & 1 & 0 & 0 \\ -0.9060 & 0 & 0 & 1 & 0 \\ -0.7862 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.68}$$

Equation (3.68) shows that commands to the four functional actuators get passed through unchanged (as represented by the diagonal ones in $\mathbf{D}_{ai}$), but that the first column redistributes the left stabilator command to the other functional actuators. Note also that the diagonal term corresponding to the assumed failure ($\mathbf{D}_{ai}(i,i)$, or $\mathbf{D}_{a1}(1,1)$ in this example) is zero, ensuring that no command is sent to the failed actuator. This form of the redistribution matrix is convenient in that, knowing the assumed failure, i, and the associated column of $\mathbf{D}_{ai}$ is sufficient to characterize the redistribution matrix completely. For ease of implementation then, a packed $\mathbf{D}_a$ matrix is created where the i^{th} column of $\mathbf{D}_a$ equals the i^{th} column of $\mathbf{D}_{ai}$. The $\mathbf{D}_{ai}$ matrix used in this research is presented in Section 5.3.3.

3.9 Chapter Summary

This chapter introduced the VISTA F-16 and the nonlinear SRF VISTA simulation that was used as the truth model. This was followed by an analysis of the linear model provided by the SRF VISTA simulation. Additional models, for the measurements, outputs, noise, and actuators, were presented which allowed the linear model to be expanded into a design model adequate for the synthesis of the LQG controllers to be embedded into the MMAC structure. The controller synthesis began with a derivation of the linear quadratic regulator problem. The motivation for Type I tracking properties led to the augmentation of pseudointegrals, and the resulting proportional plus integral controller was massaged back into the form of the LQ regulator to simplify controller synthesis. The implicit and explicit model following techniques were then presented with the premise that, if needed, they could be incorporated to help match the structure of the Block 40 flight control system and to enhance tracking and robustness. Finally the control redistribution for MMAE-based controller was developed. The next chapter will present the results of implementing the MMAC and MMAE-based control redistribution on the VISTA F-16 simulation.

4. Results

4.1 Chapter Overview

This chapter begins with an interpretation of the probability summary plots used to present the multiple model algorithm performance resulting from various tunings of the Kalman filters within the algorithm's structure. Section 4.3 then presents several attempts at using tuning to reconcile the difference between the linear and nonlinear models for lateral acceleration, followed by a presentation of the final tuning used for the MMAC in this research. Necessary modifications made to the dither signal are presented in Section 4.5, and an analysis of the numerical problems associated with a discrete-time LQG/PI controller is given in Section 4.6. Next is a presentation of the performance of the MMAE-based control redistribution method, followed by a chapter summary.

4.2 Interpretation of the Probability Summary Plots

The most vital part of any multiple model structure is the ability to provide accurate state estimates and to determine the best parameter estimate quickly. These abilities are rooted in the tuning of the Kalman filters. Therefore the first task is to ensure that the Kalman filters, and the resulting MMAC, are properly tuned. In the context of this research, a well-tuned MMAC is one which detects all failures as quickly as possible without introducing false alarms. This criterion is perhaps best evaluated through the use of probability plots. Recall that each filter and its associated failure hypothesis gets assigned a probability between zero and one, where a probability greater than 0.95 triggers a failure declaration. This declaration results in a bank swap within the hierarchical structure of Section 2.4.5, and it also is used for the decision on how to redistribute control for the MMAE-based controller presented in Section 3.8. By simultaneously viewing the probabilities associated with each filter, one can, at a glance, determine the declared failure status of the MMAC. Additionally, viewing where the probabilities are when an incorrect declaration is made gives insights as to how to tune the system for better performance.

An example of a probability plot is shown in Figure 4.1, which was produced from ten Monte Carlo runs of a truth model experiencing a normal acceleration (a_n) sensor failure at one second into the simulation. The probability plot shows twelve plots of probabilities versus time, corresponding to the twelve filters (and their associated hypotheses) in the filter bank. The y-axis label of each plot identifies the hypothesized failure condition for that particular filter, with the nomenclature defined according to Table 4.1. The plots are arranged such that they directly correspond to the ordering of the filters within the actual MMAE, and so the reader might imagine that this is what one would see if one were able to open up the MMAE and peer inside, simultaneously viewing the probability assigned to each filter at once. Each plot gives the mean (solid line) and $\pm$ one standard deviations (dotted lines) of the ten Monte Carlo runs. Because probabilities are limited to the range $0 \leq p_k \leq 1$ (actually the lower bounding described in Section 2.4.1 limits this range even further to $0.001 \leq p_k \leq 0.989$), the y-axis is limited to show only the interval between zero and one, cropping some standard deviation lines, but with no loss of information.

Table 4.1 Definition of the Y-axis Labels on Summary Plots

Abbreviation	Hypothesis	Abbreviation	Hypothesis
ff	Fully Functional Aircraft	aoa	Angle-of-attack Sensor
ls	Left Stabilator Failure	q	Pitch Rate Sensor
rs	Right Stabilator Failure	a_n	Normal Acceleration Sensor
lf	Left Flaperon Failure	p	Roll Rate Sensor
rf	Right Flaperon Failure	r	Yaw Rate Sensor
rud	Rudder Failure	a_y	Lateral Acceleration Sensor

Viewing along a line drawn vertically anywhere between the zero and one second marks of Figure 4.1 shows that a probability of approximately one has been assigned to the fully functional filter's hypothesis, while all of other filters have assigned probabilities of approximately zero. There is some slight jitter in the probabilities, but the MMAC maintains lock on the correct hypothesis. At this point the MMAE is operating correctly since the mean probability is above 0.95 and each run is initialized with a fully functional aircraft. A similar analysis made at three seconds shows that an a_n sensor failure is declared, with a transitional period evident between approximately

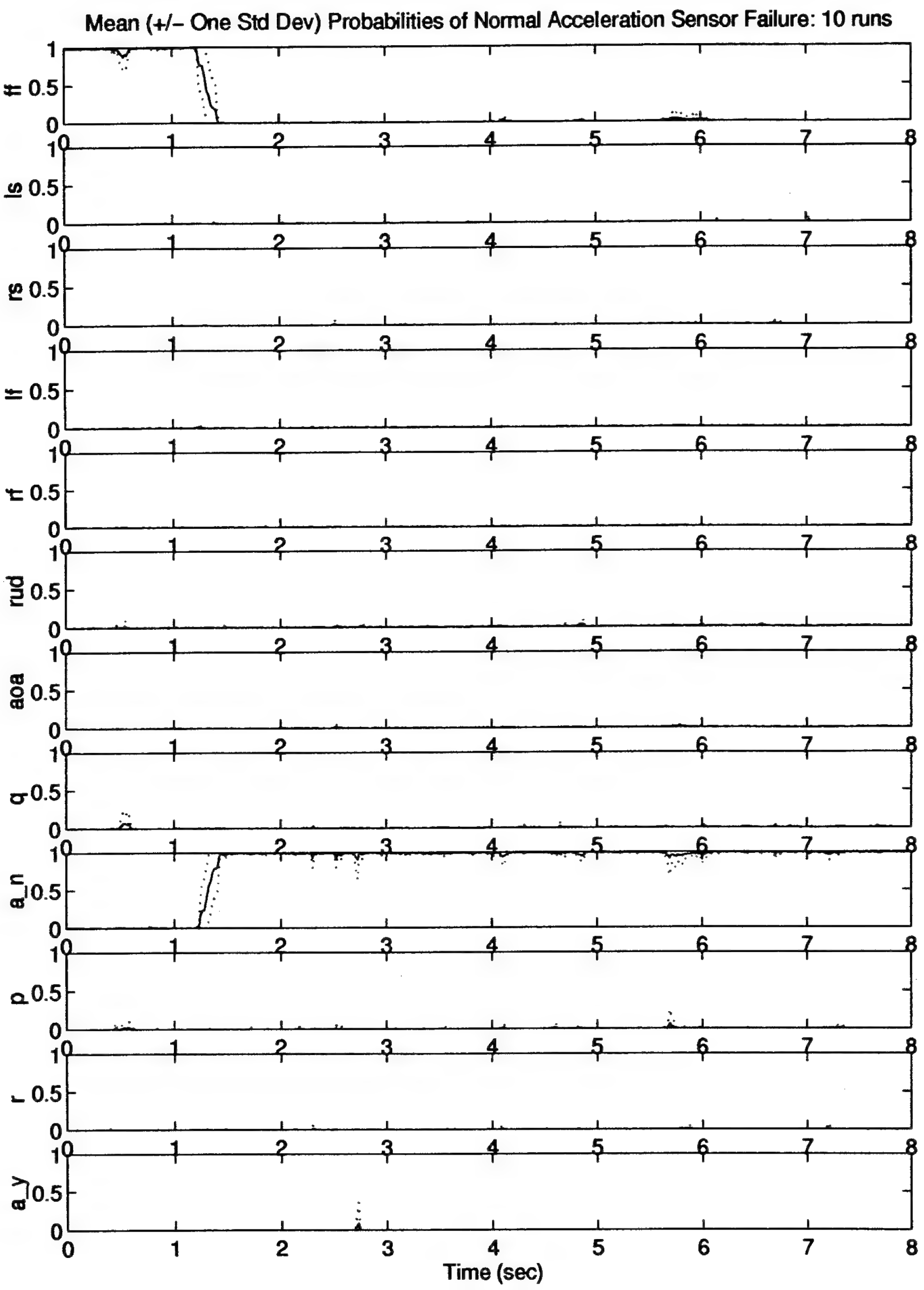


Figure 4.1 Example Probability Plot - Normal Acceleration Sensor Failure

1.25 and 1.5 seconds. As the title at the top of the figure indicates, a normal acceleration sensor failure really was implemented (at one second), so that in this example the MMAC successfully and quickly declared this failure.

Two observations can be drawn from a closer inspection of Figure 4.1. First, outside of a short transitional period, the MMAC is able to lock onto the correct hypothesis. In fact, this probability plot would seem to indicate that this is a well-tuned MMAC, though the remaining failure conditions must be tested to be certain. While there are some minor spikes in the standard deviations, indicating that during some of the Monte Carlo runs, the MMAC is experiencing unexpected motion, possibly due to unexpectedly large wind gusts, the MMAC still maintains lock on the correct declaration. The possible exceptions to this positive lock statement are the flaperons and stabilators. Being symmetric control surfaces, experience has shown that the MMAE typically has difficulty identifying the correct failure within a pair, e. g. , it can quickly declare that a stabilator failure occurs, but has more difficulty determining whether it is the right or left stabilator that has failed. On the corresponding probability plot, shown in Figure 4.2 , this is seen by a mild bouncing of the probabilities between the two flaperons. Second, even during the transitional periods, the MMAE moves smoothly from the first hypothesis to the second hypothesis, i. e. , during the transition the probabilities do not stray to any of the other filters. Based on these observations, a simplification can be used to reduce the amount of data that needs to be presented at once. Figure 4.3 shows just the strip from the probability plot containing the correct failure hypothesis from Figure 4.1. The assumption is made (and holds true for a well-tuned MMAC) that any probabilities not in this strip are contained in the fully functional filter. Therefore, it is possible to arrange twelve of these strips into a single summary plot, showing virtually all pertinent information about the MMAC's failure detection performance.

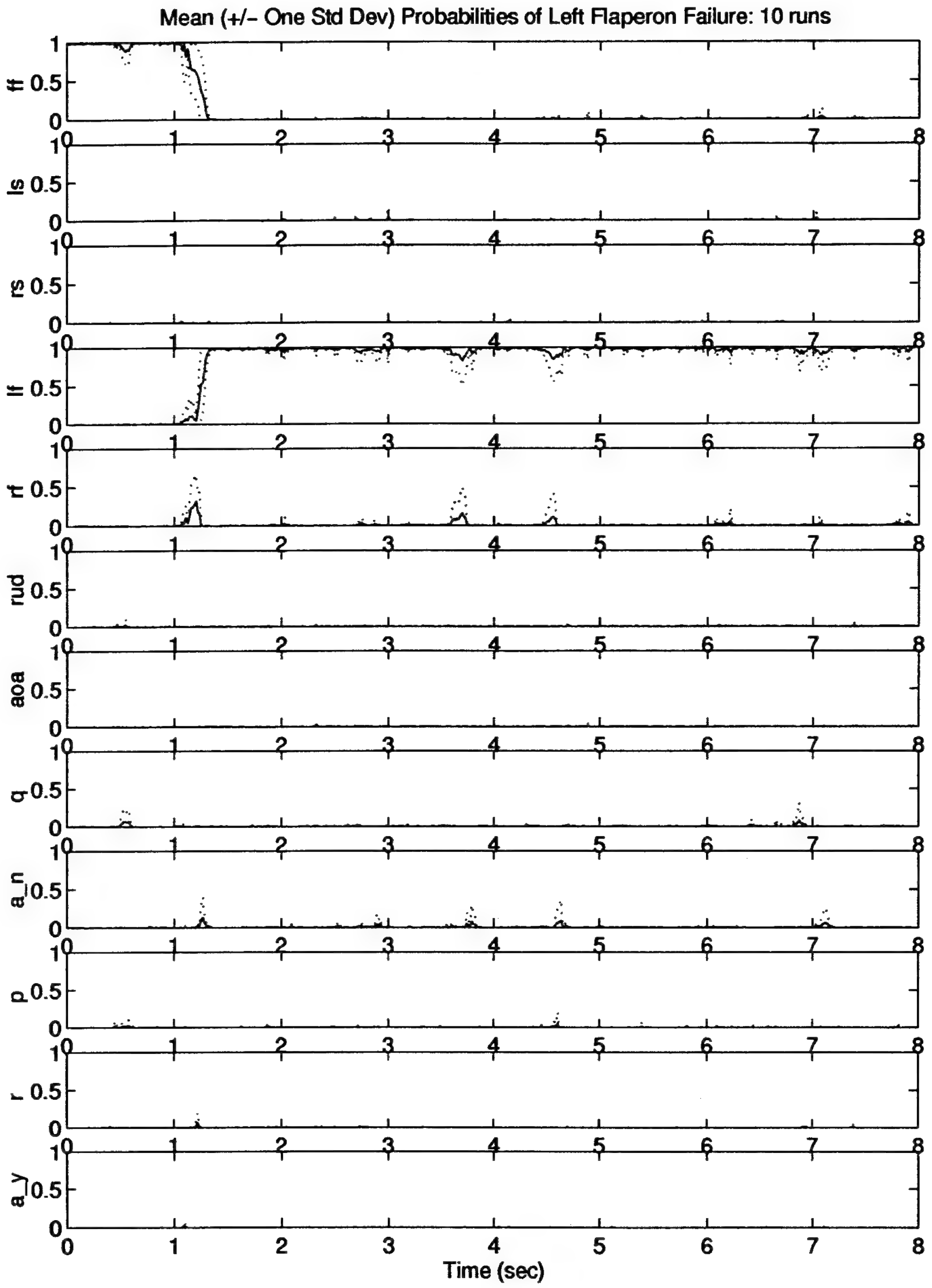


Figure 4.2 Example Probability Plot - Left Flaperon Failure

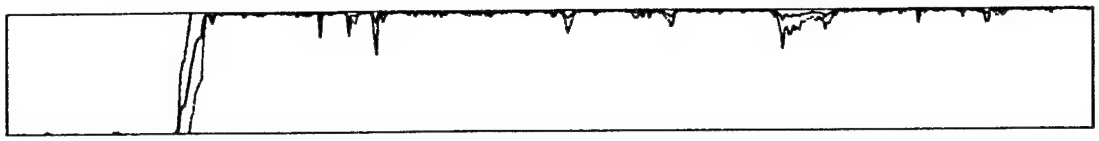


Figure 4.3 Strip Showing Only the a_n Filter's Probabilities

Figure 4.4 shows an example summary plot. Note that the only important facts to pull out of the summary plot is that the probabilities linger around zero or one except for the short transition period, marked by a sharply rising line which indicates the speed of response. Summary plots will be used for the remainder of this chapter to present the effects of tuning in a concise, but informative manner.

4.3 Lateral Acceleration Model

As revealed in Section 3.4.2, the nonlinear calculation of lateral acceleration gave a different result in both magnitude and phase when compared to the acceleration calculated by the linear model. When previous research [8] encountered the same problem, and no error in the linear model could be determined, the decision was to substitute the linear models for both normal and lateral acceleration into the nonlinear truth model. This section details an attempt at accounting for the modeling discrepancy without resorting to modifying the truth model by making such a substitution. The methods and results shown here may well extend to examples of mismodeling in other applications.

Three approaches are presented. The first two retain the nonlinear model, and attempt to compensate through tuning. The last method also retains the nonlinear model, but investigates the effects of removing the linear measurement model from the MMAC completely. In both cases, the tuning of the MMAC is accomplished by adding pseudonoise to the noise covariance matrices $\mathbf{Q}_d$ and $\mathbf{R}$. Increasing $\mathbf{Q}_d$, the dynamics noise covariance, tends to help mask mismodeling and places less emphasis on the internal model when constructing the state estimate. Increasing $\mathbf{R}$, on

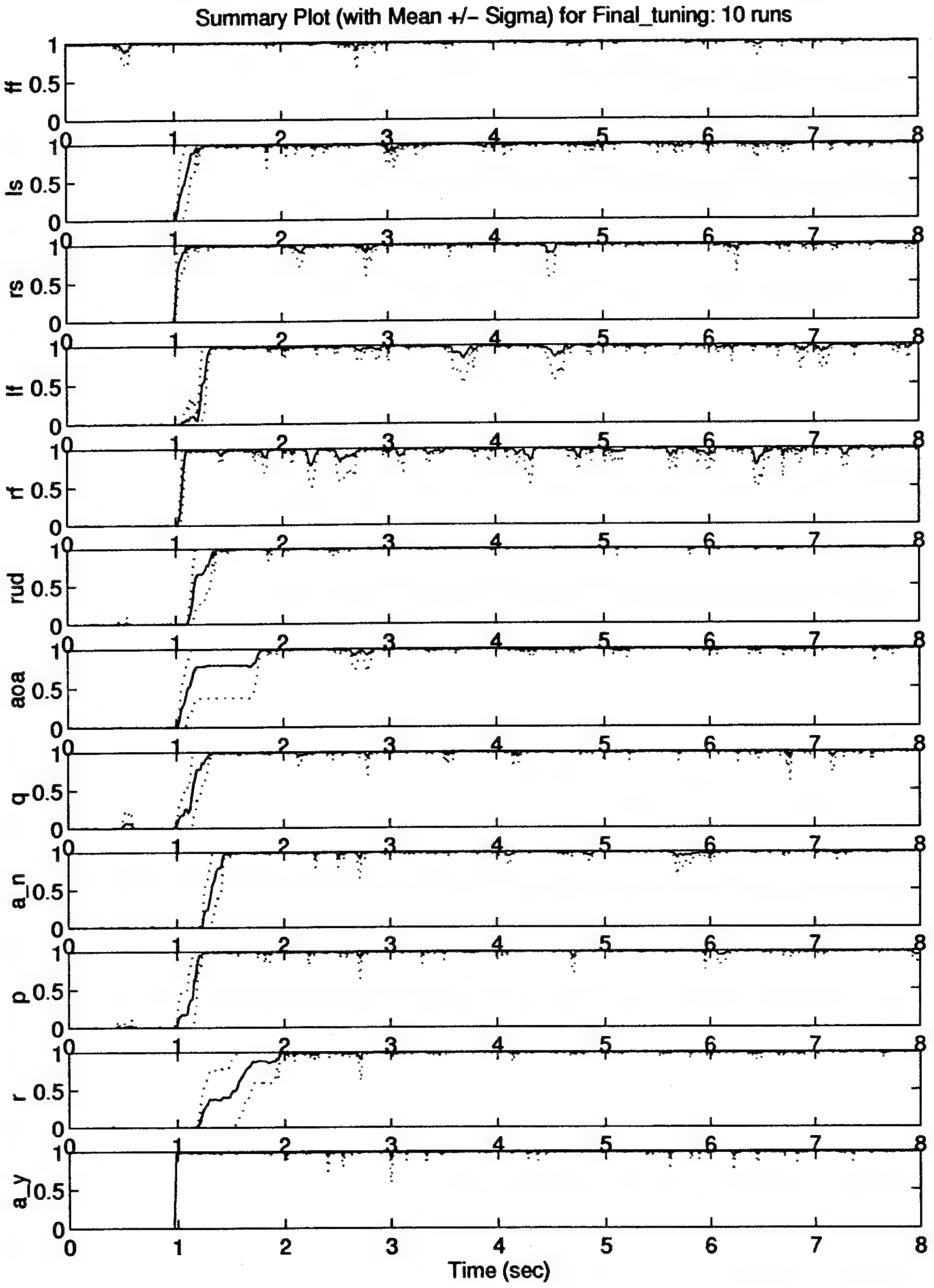


Figure 4.4 Example Summary Plot

the other hand, has the effect of decreasing the reliance on the measurements with respect to the internal model.

4.3.1 Tuning via Pseudonoise. Figure 4.5 shows the summary plot corresponding to the situation where the nonlinear lateral acceleration model is included in the truth model and no pseudonoise has been added yet. The MMAE is only able to lock on to the flaperon and a_y sensor failures. A look at two representative probability plots, shown in Figures 4.6 and 4.7, shows that the probabilities tend to bounce between the actuator hypotheses and the a_y sensor hypothesis. The reason is that the large mismatch between the two a_y models generates such a large scalar a_y residual that it dominates the residuals associated with the true failure status. The sensors, other than the a_y sensor, are not particularly involved in this phenomenon since they each have their own scalar residual, so the probabilities tend not to bounce into their associated filters. Actuator failures, on the other hand, have no single corresponding scalar residual, and, in fact, affect all the residuals through the dynamics model, so the large a_y residual pulls probabilities into their associated filters. Through tuning on the a_y entry of $\mathbf{R}$, detection is greatly enhanced for most failure conditions. Unfortunately, as shown in Figure 4.8, the improvement comes at the cost of completely forfeiting the ability to detect any true failures in the a_y sensor.

Since informing the system that the linear model should be trusted more than the nonlinear did not yield a particularly well-performing MMAE, other tuning strategies were sought. Note that reducing a noise covariance is not typically well motivated as it implies that one can achieve a better accuracy (lower error variance) than what the real-world system actually generates. Generally an alternate route of increasing $\mathbf{Q}_d$ is pursued instead. However, the exact relationship between $\mathbf{Q}_d$ and $\mathbf{R}$ is hard to define, and because a very specific response is desired, namely to allow the filters to rely more heavily on the a_y measurement only, reducing $\mathbf{R}$ was used as a technique to tune the system. The result, shown in Figure 4.9, is even worse than the result achieved by adding pseudo-

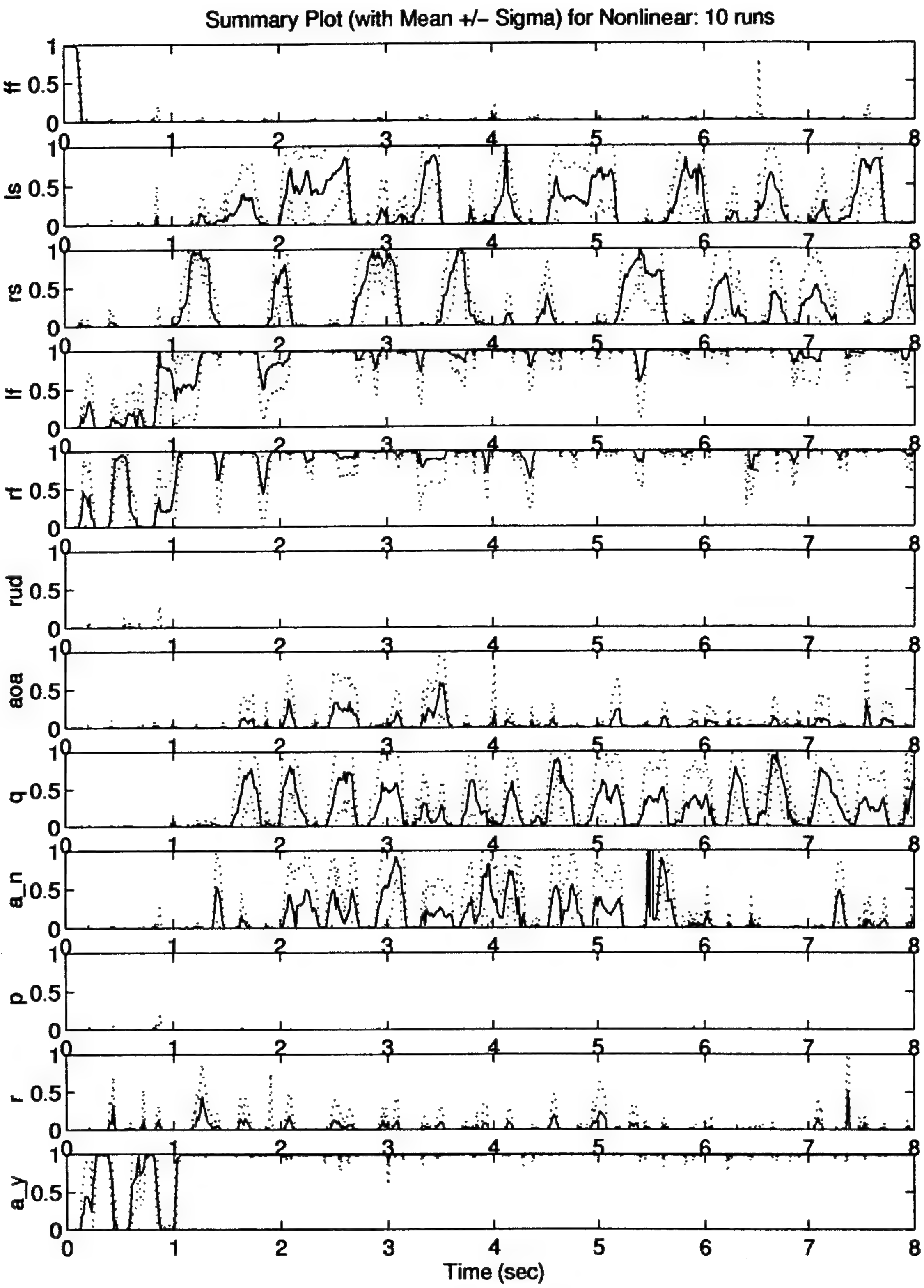


Figure 4.5 Retaining the Nonlinear a_y Calculation

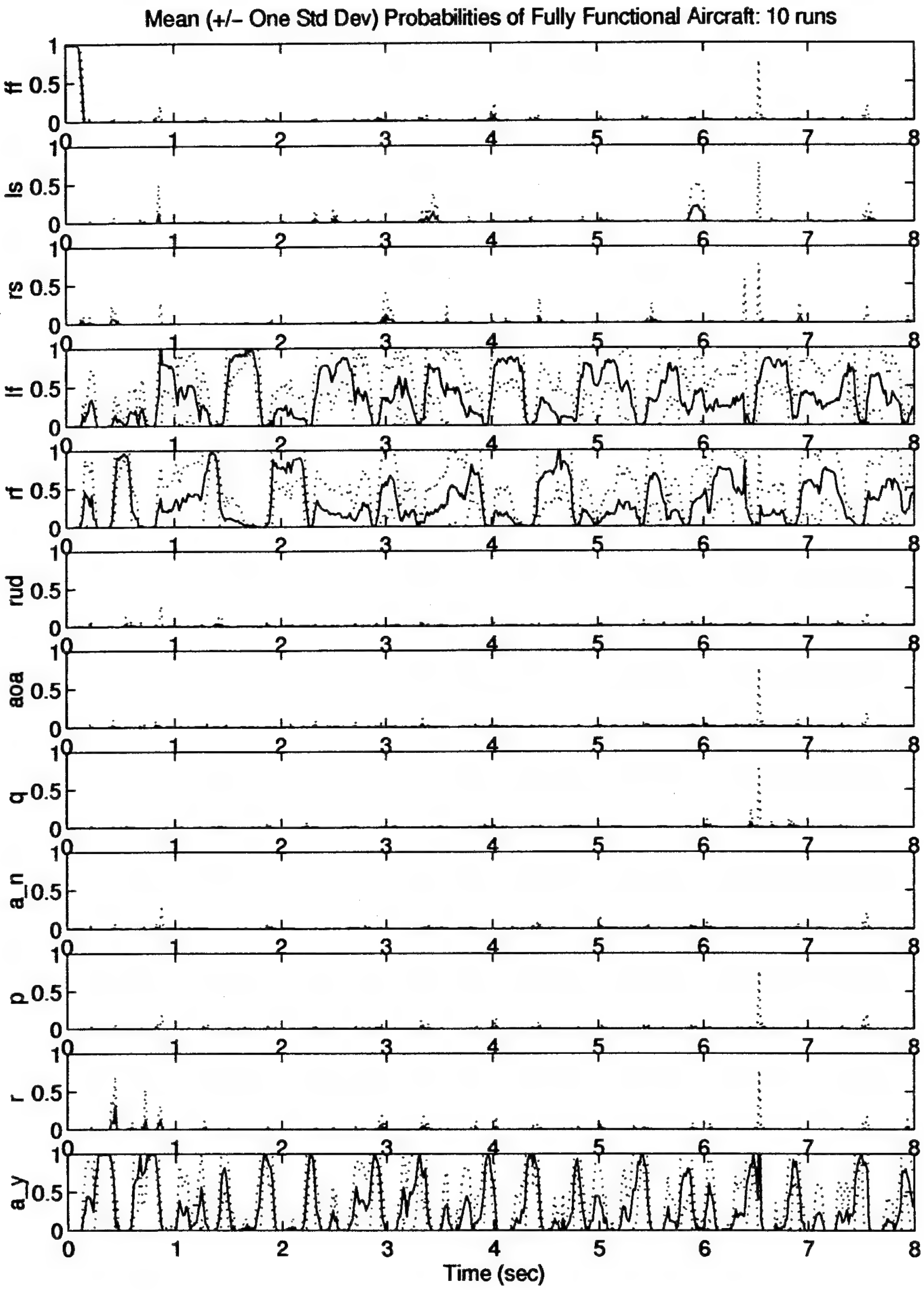


Figure 4.6 Retaining the Nonlinear a_y Calculation - Fully Functional Probability Plot

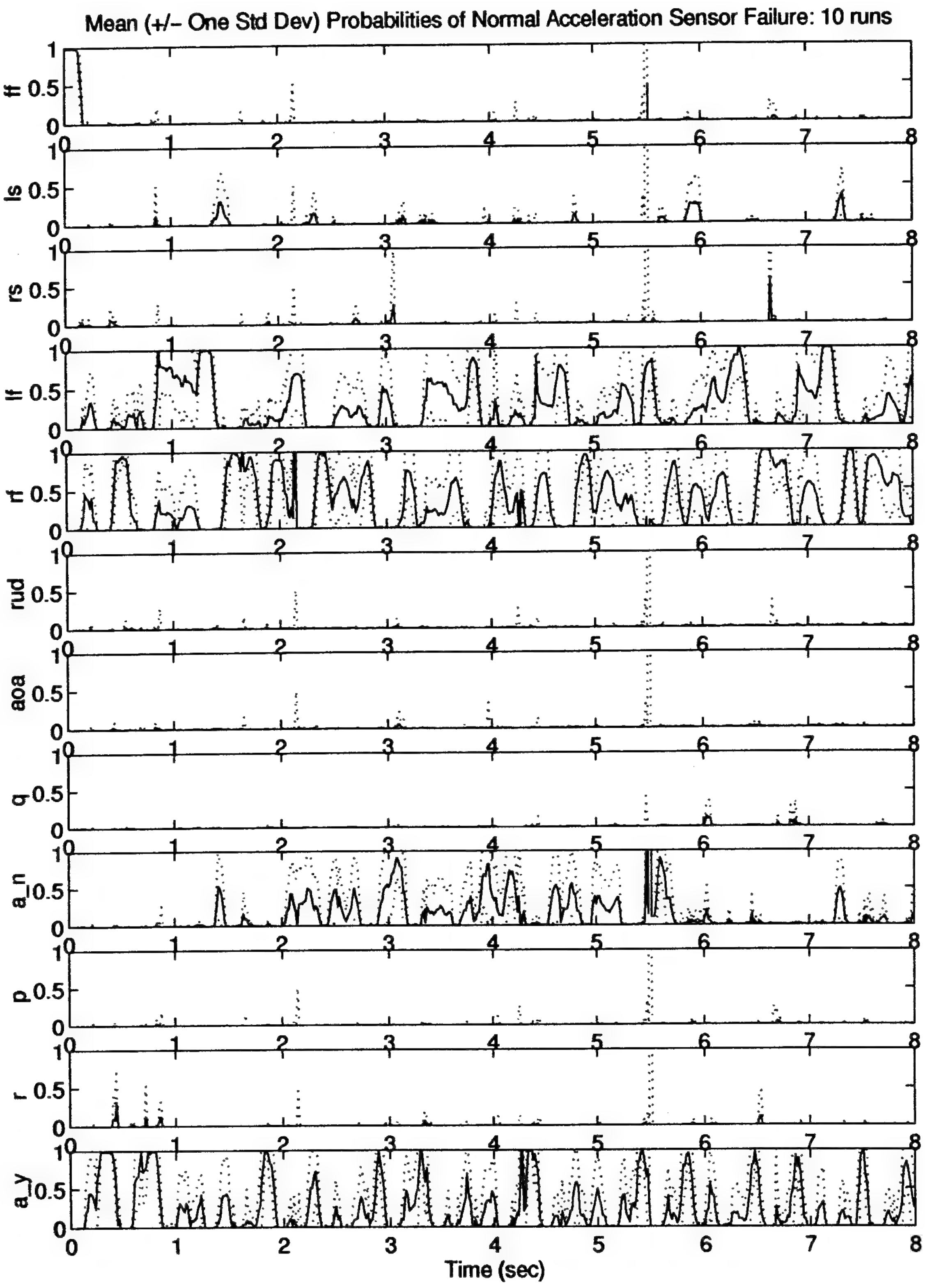


Figure 4.7 Retaining the Nonlinear a_y Calculation - Normal Acceleration Sensor Failure

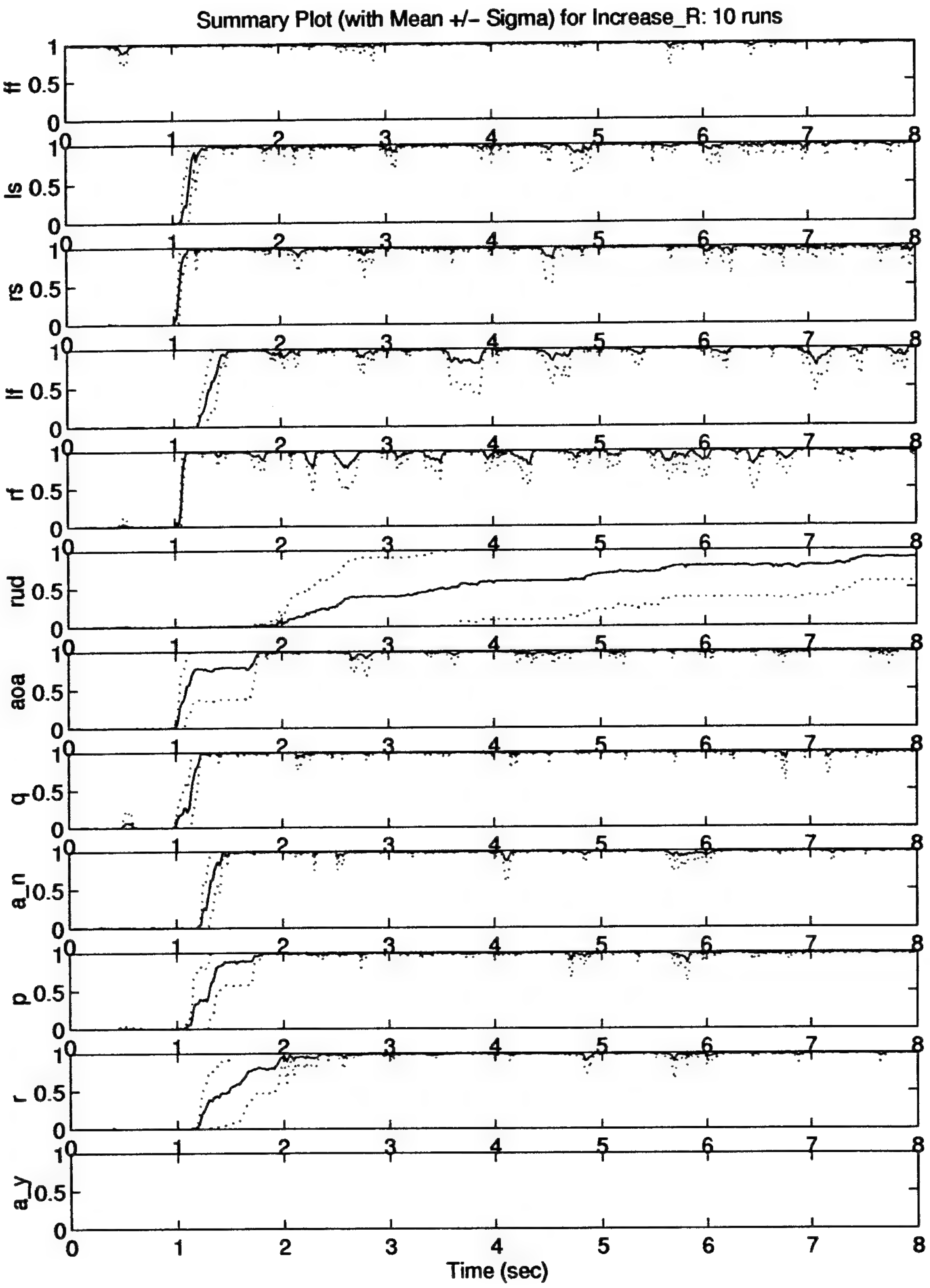


Figure 4.8 Retaining the Nonlinear a_y Calculation - Adding a_y Pseudonoise

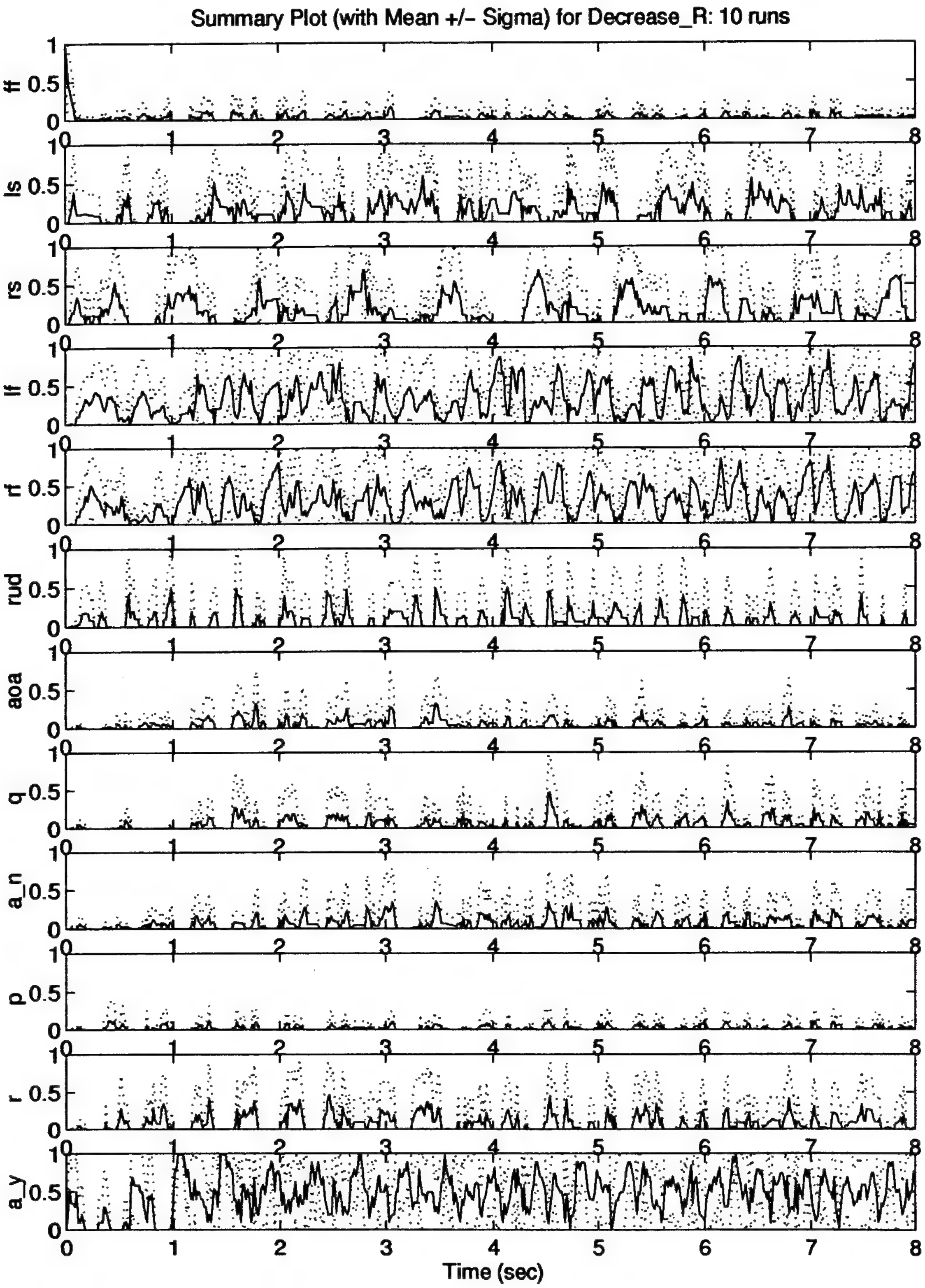


Figure 4.9 Retaining the Nonlinear a_y Calculation - Reducing the a_y RMS Noise Level

noise. Now the MMAC is unable to detect any failures whatsoever. Again, most of the probabilities are harbored by the actuator hypotheses and the a_y sensor hypothesis.

The results of tuning with the nonlinear a_y model can be summarized as: (1) No amount of tuning yielded an MMAE algorithm that was capable of detecting all failures; (2) emphasizing the linear model yielded a design with satisfactory detection of all failures not associated with the yaw channel, i. e. the rudder and a_y sensor failures; and (3) emphasizing the nonlinear model within the SRF "truth model" simulation yielded a design incapable of detecting any failures.

4.3.2 Removing the Lateral Acceleration Measurement. Based on these three observations, the linear model would appear more likely than the nonlinear one to describe the relationship of a_y with the other system variables accurately, motivating the substitution of the linear model into the SRF VISTA simulation. Before doing so, however, a second attempt to synthesize a viable MMAC without modifying the software is made. This time, the a_y sensor is completely removed from the measurement vector. In doing so, the associated scalar residual which had been overshadowing the pertinent residuals is removed. By completely removing the sensor reading from the measurement vector, the MMAC forfeits the ability ever to detect an a_y sensor failure. The resulting summary plot is given in Figure 4.10. As seen, the MMAC is no longer able to lock on to either yaw rate sensor failures or rudder sensor failures consistently. Apparently, the lateral acceleration measurement is critical to failure detection in the yaw channel, and no amount of tuning is able to improve the system.

4.4 Final Tuning of the Kalman Filters

In the process of retuning the Kalman filters, several modifications were made to the existing SRF VISTA simulations. All modifications entailed a complete retuning of the Kalman filters, demonstrating at first what appeared to be a very high sensitivity of the MMAC performance to exact tuning levels. Fortunately, the sensitivity proved to be directly correlated to the amount of

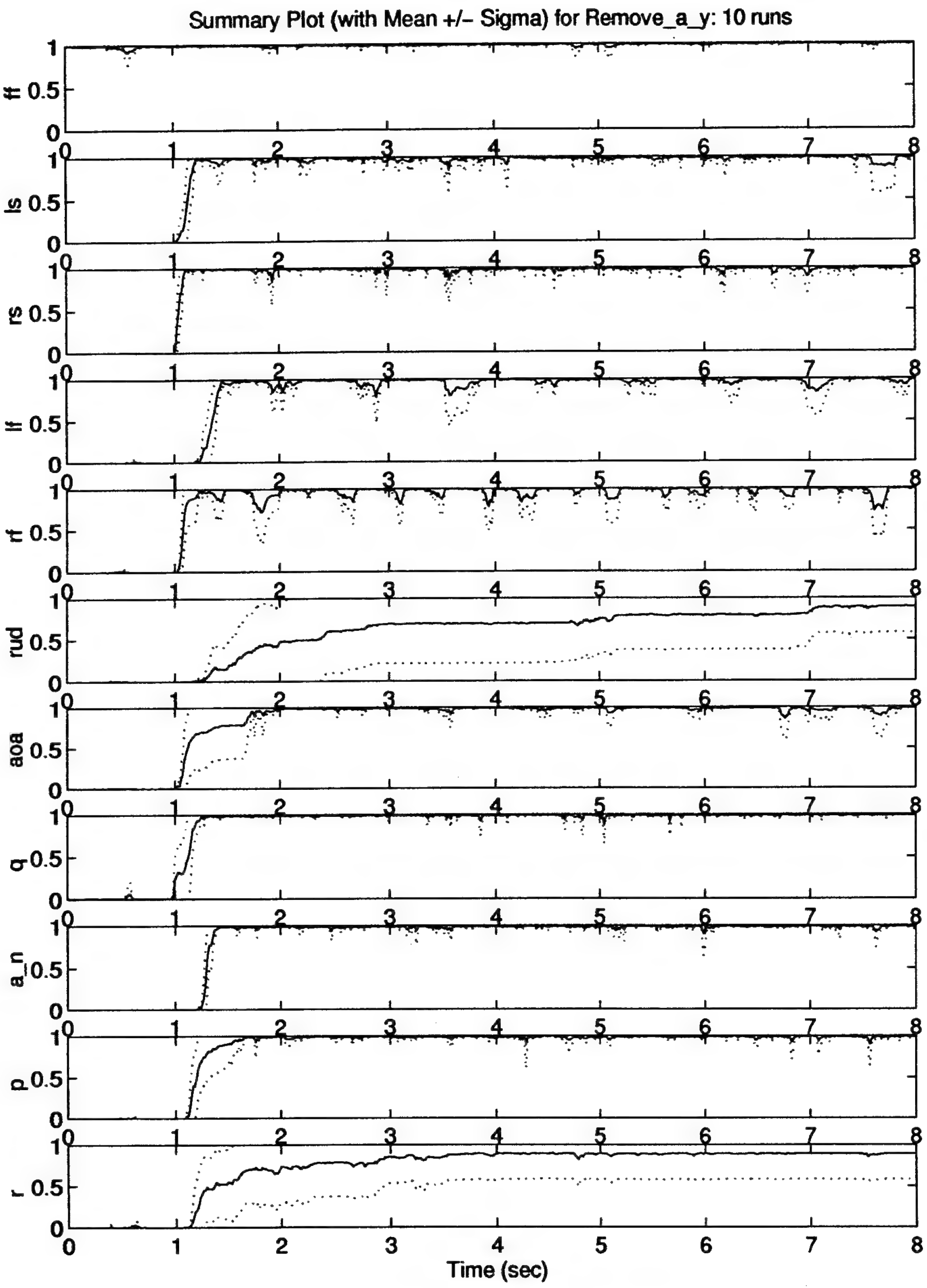


Figure 4.10 Removing the a_y Measurement

mismodeling within the SRF simulation, and the final tuning used in this research did not exhibit the same sensitive characteristic. In fact, whereas the original MMAE, from Eide's work [8], required extensive tuning on five terms within the dynamics noise, the final MMAC required no change at all from the tuning originally accomplished by Menke [34,35] against a basically linearized truth model of the VISTA F-16 (though including saturation effects) rather than a full nonlinear simulation of the real-world aircraft (as accomplished herein, using the SRF simulation). Figure 4.11 shows the performance of the MMAC with this specific tuning.

4.5 Asymmetric Dither

In the past, isolation and declaration by the MMAE of rudder and flaperon failures has been much slower than the declaration of stabilator or sensor failures [8,34]. Furthermore, although the VISTA F-16 is symmetric about the x-z plane (where the positive x-axis points out the nose, the positive z-axis is normal to the x-axis and points out the underside of the fuselage, and the origin is at the center of gravity), the right stabilator is typically harder to isolate than the left stabilator. For example, Menke showed in [34] that, although failures of the left stabilator and all sensors except the yaw rate sensor are detected in under 0.25 seconds, the MMAE was not able to isolate and declare flaperon and rudder failures until approximately 1 second. Similarly, Eide showed in [8] that the MMAE took two seconds to isolate and declare the rudder failure, and one second for the flaperons (though the right flaperon never quite locked on).

This lackluster performance shown by Menke and Eide was rationalized by the belief that the rudder and flaperons are harder to detect because they have less control authority than the stabilators [33]. Also, a contributing factor is that sensor failures are faster to detect overall compared to actuator failures because a sensor failure manifests itself directly in a single scalar residual, whereas the actuator failures appear in many scalar residuals after slowly propagating

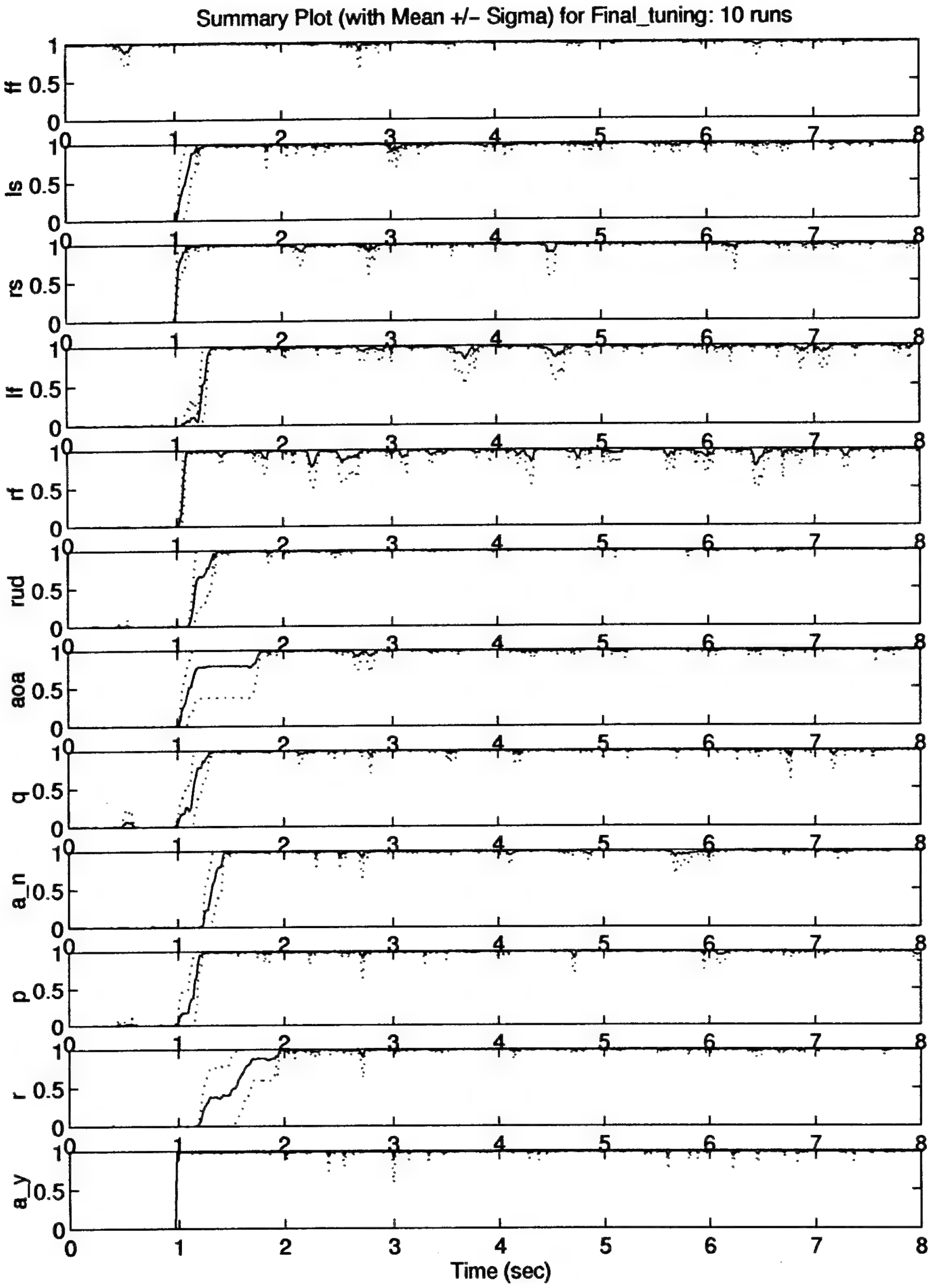


Figure 4.11 Final MMAC Performance

through the dynamics model [34]. While both arguments have some degree of validity, a close analysis, however, reveals that the slow responses are actually caused by an asymmetric dither.

The dither originally used in this research was developed by Menke [34] for use on a linear truth model of the VISTA F-16, and was carried over to the SRF VISTA nonlinear truth model by Eide [8]. The signal consists of three separate sinusoids of the same frequency but different amplitudes, applied separately to each dynamic channel of the aircraft. These dither signals, inserted via the associated input channel available to the pilot by using software to sum the dither commands with the pilot commands, are summarized in Table 4.2. Note that the pitch stick dither was required to be asymmetric so that the resulting flight path angle is constant (level flight).

Table 4.2 Original Dither Signals

Dynamic Channel	Associated Input	Command Issued	Frequency (rad/sec)	Magnitude (lb)	Phase (deg)
Longitudinal	Pitch Stick	Normal Acceleration	15	+12/-12.5	0
Lateral	Roll Stick	Roll Rate	15	11	180
Directional	Pedals	Sideslip Angle	15	30	0

This dither was originally selected based on a comparison to dithers of other wave forms [34]. Unfortunately, the interaction between the dither signals due to coupling inherent in the aircraft was not previously considered. Pronounced by the fact that the same frequency is used for all channels, the coupling effects are clearly demonstrated in Figure 4.12 in which the dither frequency is (and should be) very evident in both the pitch rate and roll rate, but is conspicuously absent in the yaw rate.

To illustrate the constructive interference occurring here, consider Figure 4.13. The first two plots show the effects on yaw rate of applying individual dither signals, on two separate runs, to the roll stick (lateral channel) and pedals (directional channel). The bottom plot shows that, if the first two yaw rate plots are summed point for point, then the resulting plot is nearly equivalent to the yaw rate resulting from applying dither to all channels at once, demonstrating destructive interference.

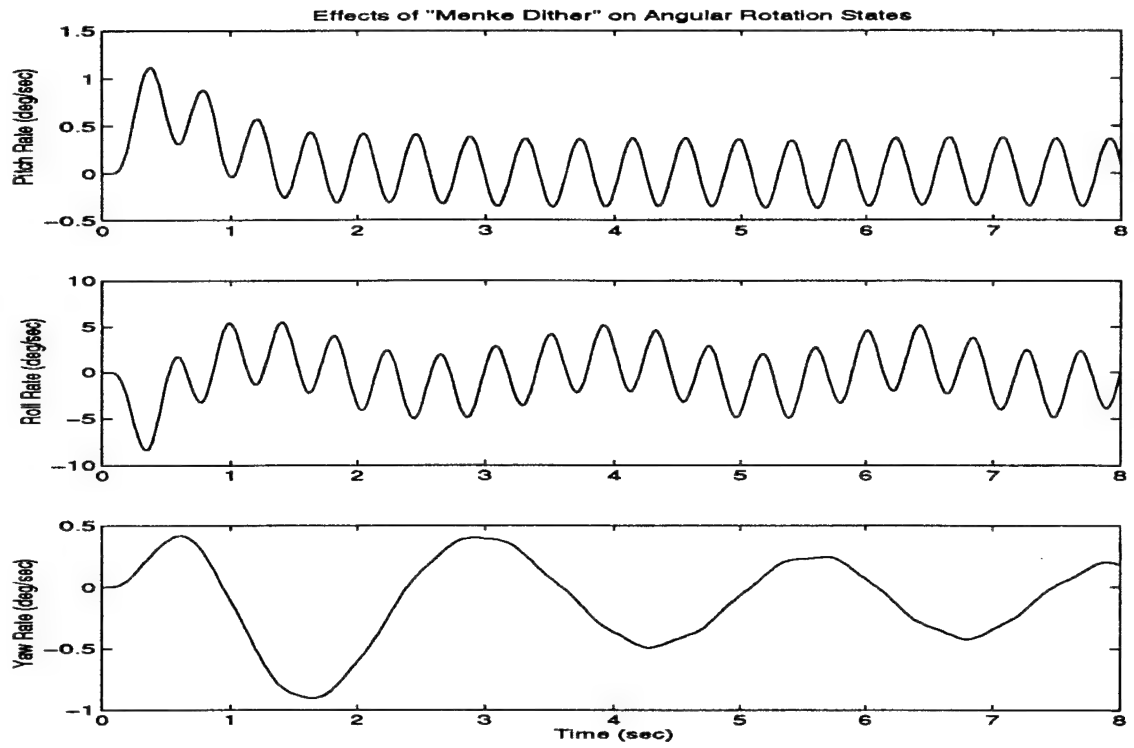


Figure 4.12 Effect of "Menke" Dither on the Angular Rates

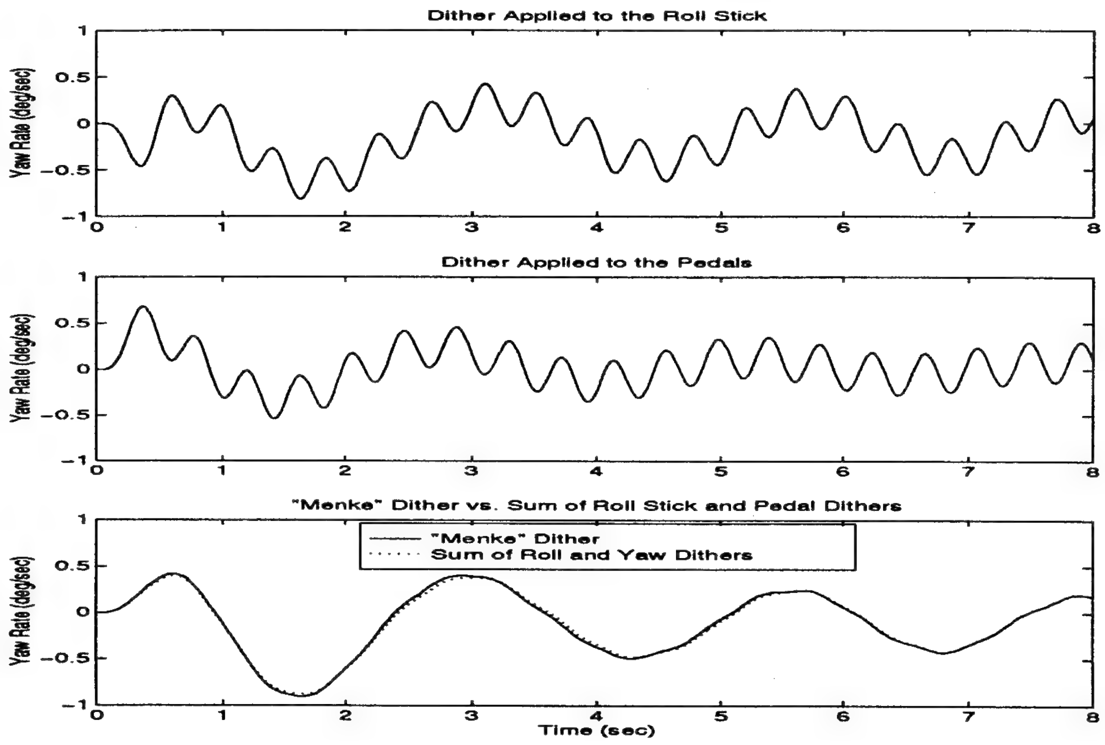


Figure 4.13 Illustration of the Interference Patterns Caused By Dither

The effects of this interference are also made obvious by looking at the five commands actually being sent to the actuators, as shown in Figure 4.14, versus the three dither commands being sent to the controller. Note the interference commands, diminished in amplitude, being sent to the right stabilator and rudder. Because failure detection heavily relies on the excitation of the states (thus the need for dither in the first place), the lack of sufficient motion of these surfaces led to the slow responses presented by Menke and Eide. Note that the offset bias in Figure 4.14 is a result of the nonzero nominal commands needed to maintain the trimmed flight condition.

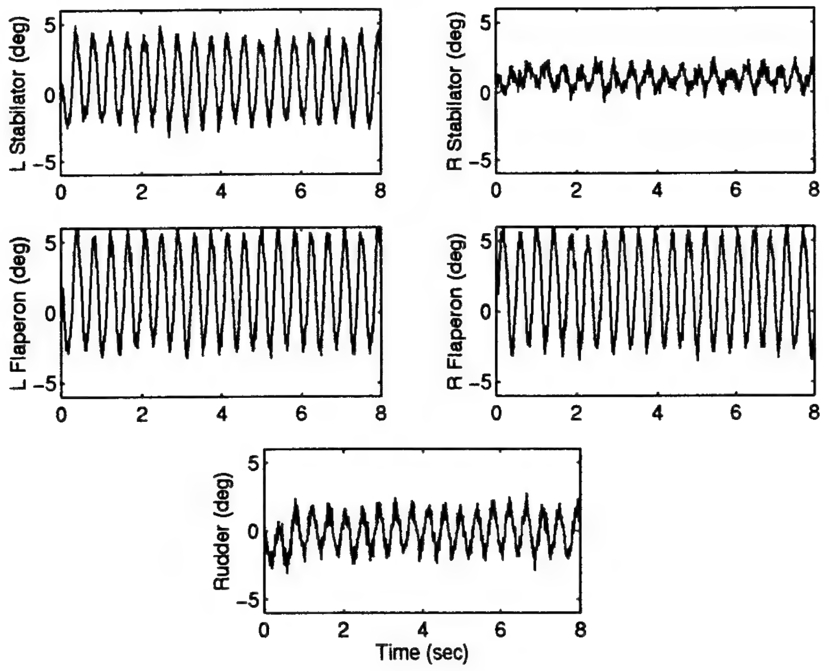


Figure 4.14 Commanded Actuator Positions Resulting From Original Dither

Three steps were undertaken in order to achieve a truly symmetric dither with respect to the actuator commands. First, the lateral dither was put back in phase with the longitudinal and directional dither signals. Although this decision was somewhat arbitrary, initial analysis showed that having the lateral and directional dither signals in phase resulted in a more effectual rudder excitation. Second, the longitudinal dither were moved out-of-phase by 90° in order to get identical peak magnitudes between the left and right stabilator. At this point the stabilators moved symmetrically (actually asymmetrically, but with equal magnitudes), but the range of motion was not enough to excite the aircraft to allow detect ability. Simply increasing the magnitude would have required an unacceptable input to the pitch stick of nearly 24 pounds, nearly all of the 30.25 pounds of input available. Therefore, the third step was to decrease the frequency of the longitudinal dither by a factor of two. The resulting commands sent to the actuators are shown in Figure 4.15 and the modified dither signals are summarized in Table 4.3.

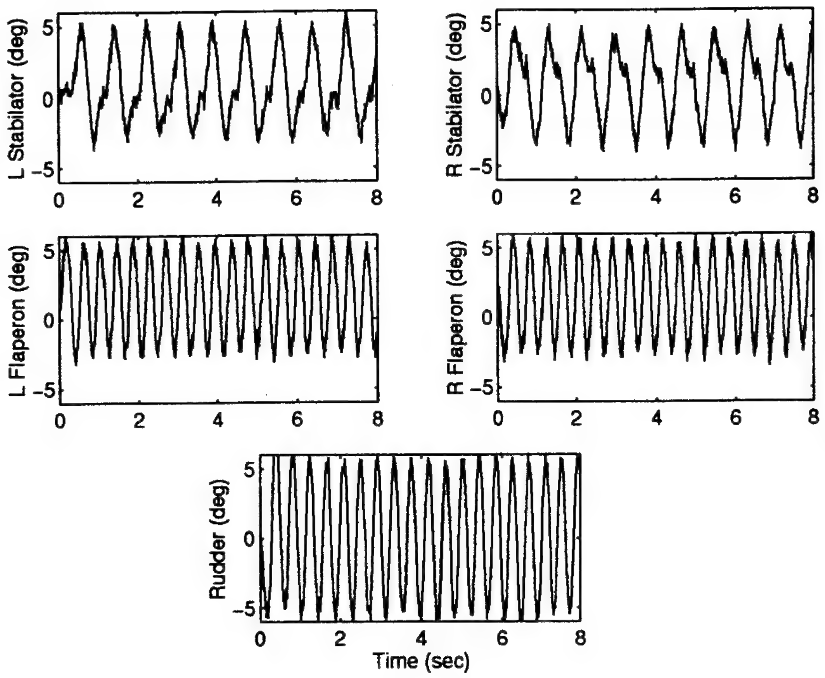


Figure 4.15 Commanded Actuator Positions Resulting From Modified Dither

Table 4.3 Modified Dither Signals

Dynamic Channel	Associated Input	Command Issued	Frequency (rad/sec)	Magnitude (lb)	Phase (deg)
Longitudinal	Pitch Stick	Normal Acceleration	7.5	+12/-12.4	90
Lateral	Roll Stick	Roll Rate	15	11	0
Directional	Pedals	Sideslip Angle	15	30	0

Although this modified dither enables the MMAC to detect all failure conditions successfully, a review of Figure 4.16, showing the trajectory of a fully functional aircraft with no commanded

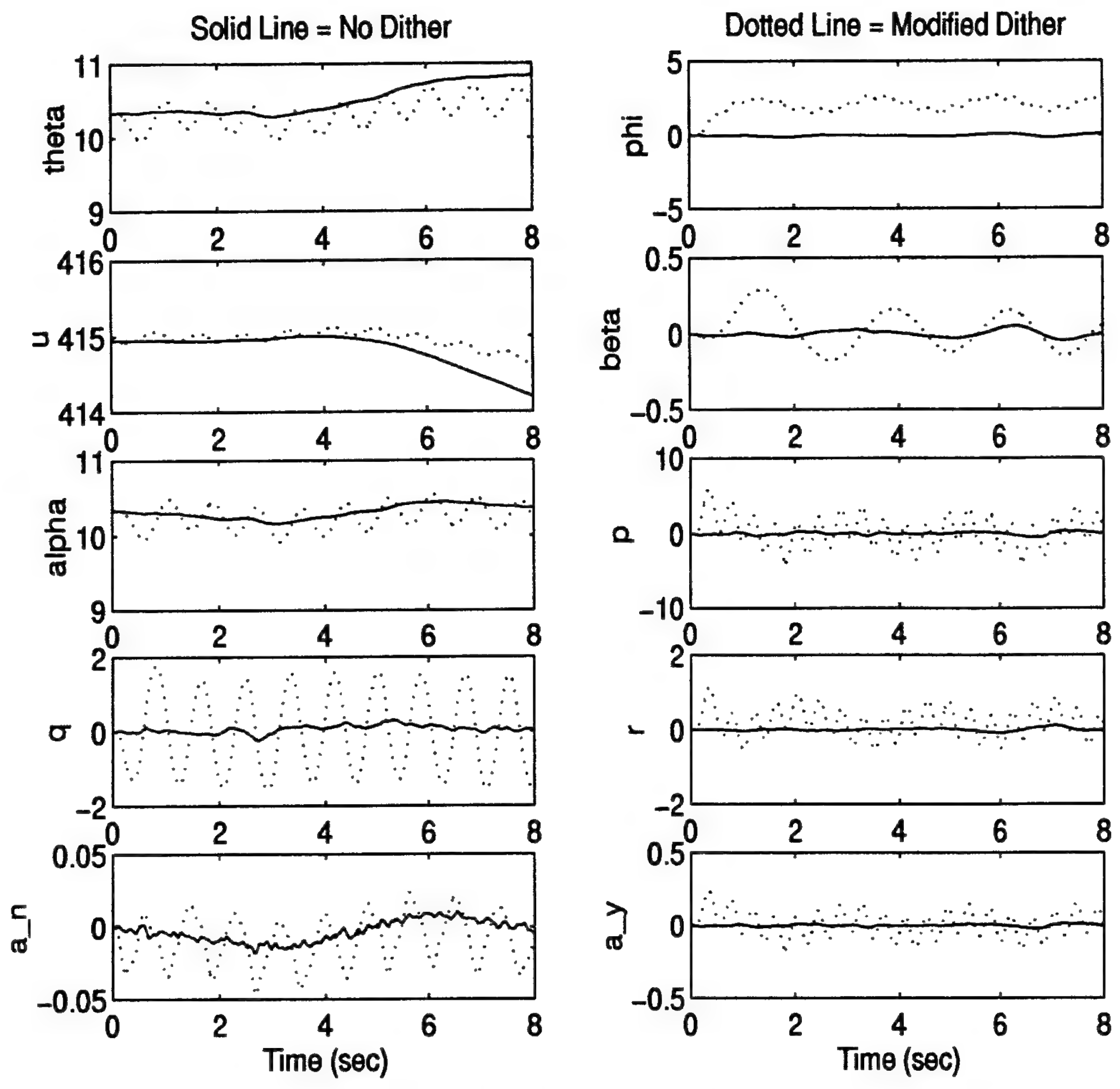


Figure 4.16 Aircraft States Resulting from Modified Dither

inputs other than the dither, reveals that while the lateral and normal accelerations still fall within specifications, the magnitude of the pitch rate oscillation calls into question the true subliminal characteristic of the dither. Another issue is that of the induced actuator rates. The highest frequency dither is at 15 radians per second. Reading the frequency off Table 4.3 and getting the approximate angular amplitude from Figure 4.15, a very rough estimate for the flaperon positions can be given as $5\sin(15t)$ degrees. Taking the derivative to get the time rate of change, the corresponding flaperon rate is roughly $75\cos(15t)$ degrees per second. Compare the amplitude of

75 degrees per second to the maximum flaperon rate given in Table 3.8 as 62 degrees per second. Fortunately, the approximation used here was conservative, and, as Figure 4.17 bears out, the dither falls nearly inside the linear region.

The fact that the dither signal consumes nearly the entire allotment of actuator rates for all surfaces may at first seem undesirable. After all, there is hardly enough control authority left for the pilot to command inputs without hitting saturations. Recall, however, that the dither signal was only necessary during benign flight where there is no pilot commanded input. Deliberate commands will in general excite the system enough to facilitate detection without the use of dither. In fact, the dither is implemented with a simple software switch that monitors the pilot commanded input and disables the dither whenever an input is detected. A pilot override to disengage the dither can also be provided.

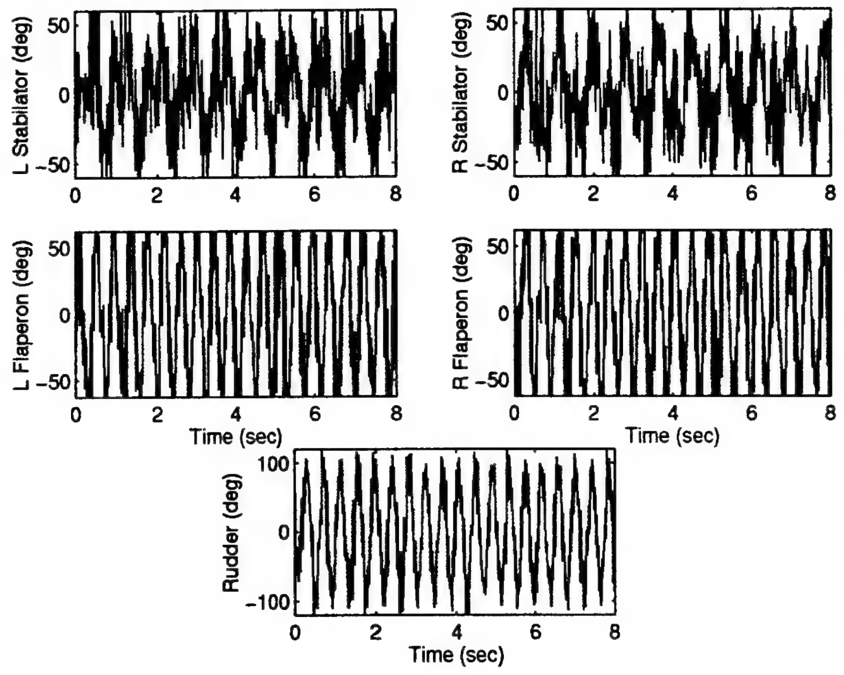


Figure 4.17 Induced Actuator Rates due to Dither

4.6 *Numerical Difficulties in the LQG Synthesis*

Having tuned the Kalman filters, the next step is to synthesize and evaluate the two control algorithms. Unfortunately, the first, the LQG synthesis, was fraught with numerical difficulties which prevented its full implementation. The source of these numerical problems is attributed to a necessary inversion in constructing the PI control law.

4.6.1 *Inverting the* Π *Matrix.*

Recall from Section 3.6.2 the steps in a PI controller synthesis. First the problem is recast as an LQR synthesis problem, using an augmented system, from which the gains, $\mathbf{G}_1$ and $\mathbf{G}_2$, can be solved. In order to achieve true Type I characteristics, however, the gains were massaged so that the system could be expressed in the form of Equation (3.41) in which feedback on the error term, $\mathbf{r}-\mathbf{y}(t_{i-1})$, forced tracking. The final result of this manipulation was a transformation as given in Equation (3.46) and repeated here:

$$\begin{bmatrix} \mathbf{K}_x & \mathbf{K}_\xi \end{bmatrix} = \begin{bmatrix} \mathbf{G}_{c1}^* & \mathbf{G}_{c2}^* \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}-\mathbf{I} & \mathbf{B}_d \\ \mathbf{C}_{aug} & \mathbf{0} \end{bmatrix}^{-1} \tag{4.1}$$

where the trailing matrix can be defined as $\mathbf{\Pi}$, and thus the augmented matrix within the inverse can be designated $\mathbf{\Pi}^{-1}$.

While generating the LQG/PI controller via a MATLAB [21] script file, it became apparent that $\mathbf{\Pi}^{-1}$ was in fact not invertible. To resolve the problem, first allow the following definitions. For the discrete-time, deterministic system,

$$\begin{aligned} \mathbf{x}(t_{i+1}) &= \mathbf{\Phi}\mathbf{x}(t_i) + \mathbf{B}_d\mathbf{u}(t_i) \\ \mathbf{y}(t_i) &= \mathbf{C}_{aug}\mathbf{x}(t_i) \end{aligned} \tag{4.2}$$

let n be the number of states, m be the number of inputs, and p be the number of outputs. The matrices in Equation(4.2) can then be stated in terms of the following dimensions:

$$\mathbf{\Phi}_{(n\times n)}, \quad \mathbf{B}_{d\ (n\times m)}, \text{ and } \quad \mathbf{C}_{aug\ (p\times n)} \tag{4.3}$$

The composite matrix, $\mathbf{\Pi}^{-1}$, is then of dimension $(n+p)\times(n+m)$, or, since the proper dimensional values are n=13, m=5, and p=3, respectively, dimension 16×18. Since $\mathbf{\Pi}^{-1}$ is not a square matrix, the best that can be anticipated is a right inverse [1]. Furthermore, the (right)inverse will only exist if $rank(\mathbf{\Pi}^{-1}) = 16$. Herein lies the problem, for the design model chosen, the composite matrix is rank deficient by one. A close inspection of $\mathbf{\Pi}^{-1}$ reveals the cause, which is not limited to just this design model or even just to aerospace applications.

Consider the first row of $\mathbf{\Pi}^{-1}$ and note the first order approximations [23] to Equations (3.20) and (3.21): $\mathbf{\Phi} \approx \mathbf{I} + \mathbf{A}_{aug}\Delta t$ and $\mathbf{B}_d \approx \mathbf{B}_{aug}\Delta t$. Using these approximations, the composite matrix can be rewritten as:

$$\mathbf{\Pi}^{-1} \approx \begin{bmatrix} \mathbf{A}_{aug}\Delta t & \mathbf{B}_{aug}\Delta t \\ \mathbf{C}_{aug} & \mathbf{0} \end{bmatrix} \tag{4.4}$$

Substituting in for the augmented matrices using the definitions implicit in Equations (3.17) and (3.19) this becomes:

$$\mathbf{\Pi}^{-1} \approx \begin{bmatrix} \mathbf{A}\Delta t & \mathbf{B}\Delta t & \mathbf{0} \\ \mathbf{0} & -14*\mathbf{I} & 14*\mathbf{I} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} \end{bmatrix} \tag{4.5}$$

[1] If a square matrix were absolutely essential at this point, then two additional, linearly independent, outputs could be created to force the number of inputs to equal the number of outputs, causing the composite matrix to be square.

where the diagonal blocks are 8x8, 5x5, and 3x3, respectively. In demonstrating rank, only the linear dependence of the rows is of interest, so multiply the rows by arbitrary constants for emphasis:

$$\Pi^{-1} \approx \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{I} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} \end{bmatrix} \tag{4.6}$$

Referring back to Equations (3.1) and (3.3), the first and last row partitions are actually the continuous-time dynamics and output models. Therefore, if any output is linearly dependent on the derivatives of the system states, then the composite matrix will be rank deficient. Consider C^*, the output used for controlling the longitudinal channel. From the derivations of the output models in Section 3.4.2:

$$C^* = g_1 \cdot A_n + g_2 \cdot q \tag{4.7}$$

For the the linear model, the derivative of pitch angle is pitch rate,[2] as is shown in the first row of the dynamics model in Equation (3.1). Substituting in this relationship along with the model for normal acceleration gives:

$$C^* = g_1 \cdot \left(-\frac{u}{g} \left(\dot{\alpha} - \dot{\theta} \right) + \frac{l_x}{g} \dot{q} \right) + g_2 \cdot \dot{\theta} \tag{4.8}$$

By rearranging terms and grouping constants, it is clearly seen that C^* is linearly dependent on three components of longitudinal state variable derivatives:

$$C^* = \left(-\frac{g_1 u}{g} \right) \dot{\alpha} + \left(-\frac{g_1 u}{g} + g_2 \right) \dot{\theta} + \left(\frac{l_x}{g} \right) \dot{q} \tag{4.9}$$

[2]Seemingly intuitive, this relationship actually only holds for wings level flight where the Euler angle, Θ [3], is the same as the pitch angle, θ.

Furthermore, analogous to the pitch rate - pitch angle relationship, roll rate for the linear model is equivalent to the derivative of roll angle, so, in fact, two of the three outputs are linearly dependent on derivatives of the input, *provided the first order approximations are valid.*

Under what circumstances do the approximations hold true? One criterion is that the sampling period be much shorter than the characteristic times that describe transient system behavior [24]. Borrowing from linear system theory, the engineering rule that can be applied is that, if the ratio of real parts of the two system poles is greater than five, then the pole with the larger real part is non-dominant [39]. Operating at 64 Hz, the sampling period of the flight control system is 15.625 msec. Comparing 0.016 to the real part of the eigenvalues (the reciprocal of the real part is the time constant of the corresponding mode [39]) will then give an indication as to the validity of the first order approximation. A summary of this analysis is provided in Table 4.4.

Table 4.4 Eigenvalues and their Ratio to Sample Period

Longitudinal Channel		Lateral/Directional Channel	
Eigenvalues	Ratio	Eigenvalues	Ratio
$-0.0268 \pm 0.1376i$	1.71	$-0.3365 \pm 2.4070i$	21.5
0.8341	53.4	-0.0444	2.84
-1.7455	111	-1.2305	78.8

Note that the linear model can be decoupled into separate longitudinal and lateral/directional dynamics, as seen in the block diagonal structure of $\mathbf{A}$ in Equation (3.1). Table 4.4 separates these two sets of dynamics in order to highlight for which channels the first order approximation can be expected to hold true. As shown, all modes are faster than 0.016 sec (though two modes do not meet the "five times greater" criteria), indicating potential numerical difficulties in both channels.

In practice, the numerical difficulties do appear first in the longitudinal channel. During the construction of the $\mathbf{\Pi}$ matrix, it was the C^* output row which caused $\mathbf{\Pi}^{-1}$ to be rank deficient (at least to the numerical accuracy of the software used [21]). Although neither a true inverse nor even a right inverse exists for a rank deficient matrix, the use of a pseudoinverse [43, 47] allowed the synthesis to continue. The pseudoinverse works by exploiting the singular value decomposition to invert the matrix where possible and by minimizing the norm-squared error for all other directions [47]. Physically, using the pseudoinverse for the LQG synthesis means that the outputs which were linearly independent in the $\mathbf{\Pi}^{-1}$ matrix will track appropriately, but that the linearly dependent rows will give only a solution with a least squares minimum error.

4.6.2 Comparison to a Previous LQG/PI Synthesis. Martin [20], who designed separate LQG controllers for the longitudinal and lateral/directional channels, had similar difficulties. In Section 4.1 of his MS thesis [20], he found the LQG synthesis yielded an unstable design if C^* was chosen as the output variable. Through a similarity transform on the design model, he was able to get what appeared to be stable output, provided pitch rate, q, was used as the output variable. [3] In fact, however, stable performance was never truly achieved, as seen in the plots of Appendix C of [20]. Martin may have been confusing the stability of controlled system with the ability of his software package (an LQ synthesis program written by Payson [40]) to complete the synthesis. Output from a sample run through this program, included in Appendix A of [20], reveals that a pseudoinverse was indeed used (note the comment on page A-33 of [20]). This comparison provides greater confidence that the current synthesis effort is being correctly performed, and that the numerical difficulties are inherent to the problem and not due to erroneous implementations.

4.6.3 Response of the LQG/PI Controller. Despite the fact that numerical problems limited the performance of the LQG/PI controller, the longitudinal response was still evaluated to determine whether or not it would be suitable for controlling the aircraft. The initial weighting matrices used to begin the iterative tuning process were created by placing the reciprocal of the maximum values squared down the diagonal. The maximum values correspond to the smallest deflection limits for $\mathbf{U}$, the rate limits for $\mathbf{U}_R$, and the maximum allowable command inputs as shown on the command gradients [11] for $\mathbf{Y}$. For reference, the deflection and rate limits are given

[3]Because C^* was specified as the controlled variable, Martin iterated using q as the output variable until satisfactory C^* performance was achieved.

in Table 3.8, and the maximum allowable command inputs are summarized in Table 4.5. The resulting matrices are:

$$\mathbf{Y} = \begin{bmatrix} \left(\frac{1}{10.86}\right)^2 & & \mathbf{0} \\ & \left(\frac{\pi}{324\times 180}\right)^2 & \\ \mathbf{0} & & \left(\frac{\pi}{30\times 180}\right)^2 \end{bmatrix} \tag{4.10}$$

$$\mathbf{U} = \begin{bmatrix} \left(\frac{\pi}{19\times 180}\right)^2 & & & & \\ & \left(\frac{\pi}{19\times 180}\right)^2 & & \mathbf{0} & \\ & & \left(\frac{\pi}{21.5\times 180}\right)^2 & & \\ & \mathbf{0} & & \left(\frac{\pi}{21.5\times 180}\right)^2 & \\ & & & & \left(\frac{\pi}{30\times 180}\right)^2 \end{bmatrix} \tag{4.11}$$

$$\mathbf{U}_R = \begin{bmatrix} \left(\frac{64\pi}{60\times 180}\right)^2 & & & & \\ & \left(\frac{64\pi}{60\times 180}\right)^2 & & \mathbf{0} & \\ & & \left(\frac{64\pi}{62\times 180}\right)^2 & & \\ & \mathbf{0} & & \left(\frac{64\pi}{62\times 180}\right)^2 & \\ & & & & \left(\frac{64\pi}{120\times 180}\right)^2 \end{bmatrix} \tag{4.12}$$

Note that the number 64 in the numerator of the $\mathbf{U}_R$ terms is the sample frequency of the flight control system and is needed to convert the rate limits, given in terms of a continuous-time derivative, into equivalent pseudorates as generated by the discrete-time system.

Table 4.5 Maximum Allowable Command Inputs

Command Gradient	Maximum Command	Units
Pitch	10.86	g's
Roll	324	deg/sec
Rudder	30	deg

Unlike previous research which designed separate longitudinal and lateral/directional controllers [20, 29, 31] the ability of the MMAC to correct for actuator failures relies partly on the inherent redundancy between control surfaces. For example, although less efficient because of a smaller moment arm, the flaperons are capable of inducing a pitching moment, a motion primarily attributed to the stabilators in normal operation. This interdependence requires a single model to be used for the synthesis. Therefore, since the control surfaces are used by all channels, the tuning on any one channel will primarily involve modifying the components of the **Y** weighting matrix which penalizes the tracking error. For example, in the longitudinal channel, the stabilators are the most effectual control surface. In tuning then, one might want to penalize more heavily the flaperons and rudder in order to minimize unwanted effects. Doing so however will degrade performance in the lateral channel since the same weighting matrices are used, and in the lateral channel the flaperons are the primary control surface. On the other hand, tuning performed on the C^* component of **Y** will greatly effect performance in the longitudinal channel with little to no adverse effect on the lateral and directional channels.

Response in the longitudinal channel was investigated to reveal the effects of the pseudoinverse. The initial closed loop system was unable to track a step C^* input command, and was also unable to settle to a steady-state value within the eight second simulation window. Although the performance would seem to indicate an unstable system, a check of the closed loop eigenvalues revealed a stable system (all eigenvalues of the state transition matrix were contained within the unit circle). Possibly the system does eventually converge to a steady-state solution, but for aircraft control, an acceptable settling time must be on the order of seconds, not minutes. In any case, a series of iterations on the weighting matrices showed no notable improvement in performance. Figure 4.18 shows the C^* step response for some representative tuning iterations on the C^* component of **Y**. Note that increasing the penalty on deviation from tracking the step input results not only in a faster rise time, but also in a faster fall off after the peak has been reached. Not shown on the graph is the fact that the bottom (dash-dot) line is something of a lower bound in that scaling $\mathbf{Y}(1,1)$ by factors less than one resulted in curves nearly the same as this one.

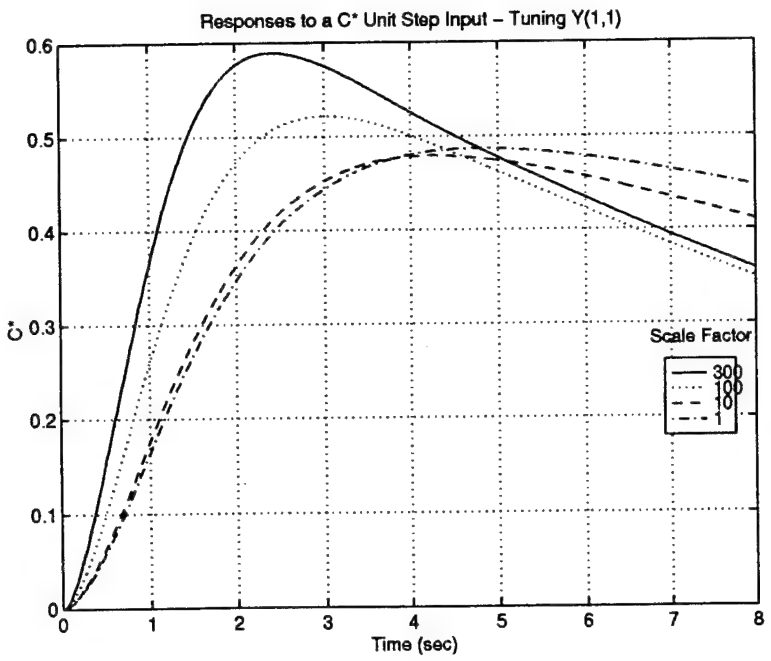


Figure 4.18 Effect of Scaling the C^* Penalty

Figure 4.19 shows that, as expected, increasing the penalty on the pseudorate greatly cripples the ability of the actuator to respond rapidly. The effect on C^* was to scale the magnitude reached by the curve, but not to alter the shape of the C^* curve. Interestingly, increasing the control input weighting matrix, **U** had the opposite effect compared to what was expected. Generally, one might expect that, as the control input was more heavily penalized, the input signal would diminish in size, and the state deviations would increase. Actually the opposite occurred, as seen in Figure 4.20. Increasing the penalty actually increased the size of both the stabilators and flaperons, as well as the output, C^*.

4.7 Matching the LQG/PI Controller to Block 40 FCS

One of the implicit assumptions made in this research was that an LQG controller could be synthesized, possibly with the inclusion of implicit or explicit modeling, which would closely

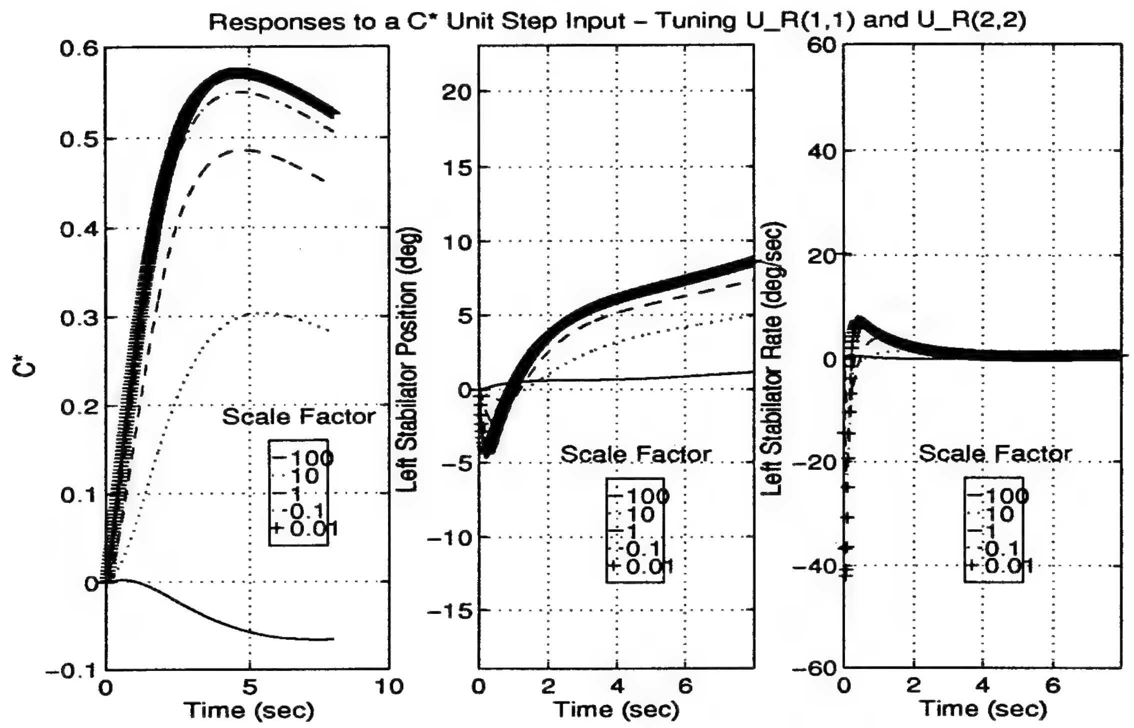


Figure 4.19 Effect of Scaling Pseudorate Penalties

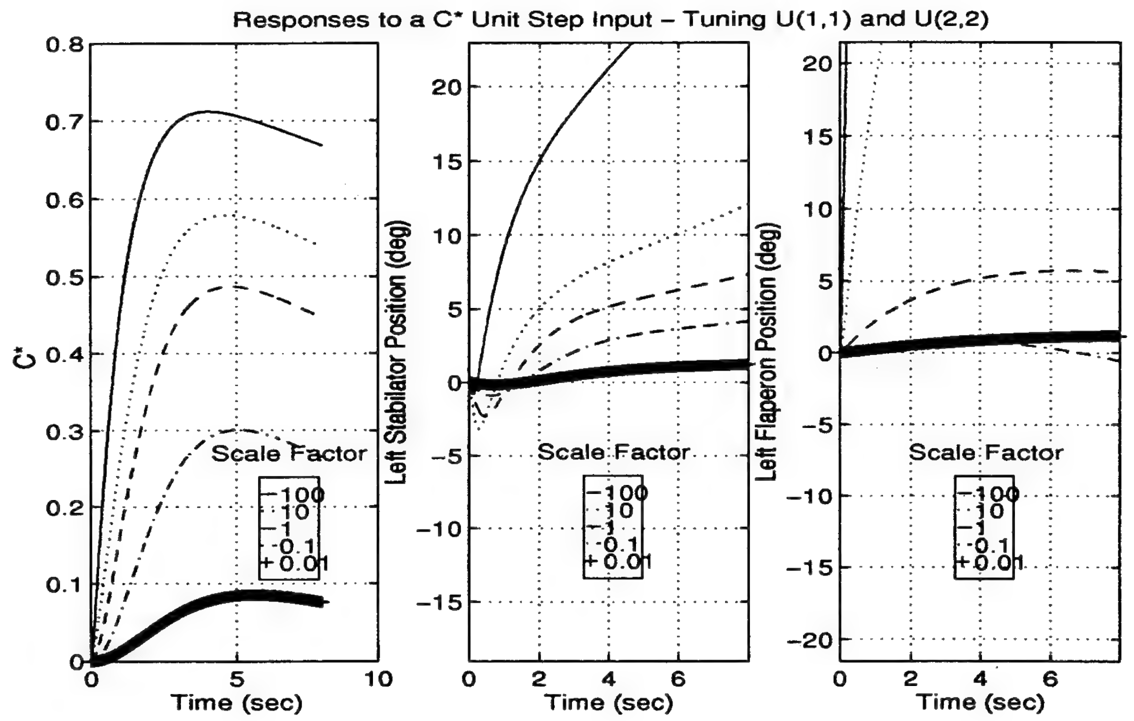


Figure 4.20 Effect of Scaling Actuator Position Penalties

match the performance of the Block 40 flight control system. In making this assumption, some fundamental differences between linear and nonlinear controllers were overlooked which would prevent this match from occurring.

Linear system theory shows that a linear system's response can always be decomposed into a homogeneous solution and a particular solution, where the homogeneous solution to differential equations can be expressed as a sum of exponential terms and the particular response is based on the applied input. Further, for a stable system, the homogeneous exponents are all negative, resulting in a zero steady-state solution. Also, for a linear system, the particular solution for a zero input is a zero output. Therefore, for a zero input, the complete solution must be zero. This fact has particular implications for this research where a doublet is used as the test signal. [4] Consider a fast pitch doublet beginning at 1 second and finishing one second later (this is the form of the test signal used in this research to evaluate tracking). After two seconds into the simulation, there is zero input and so the system states and outputs begin to decay away to zero. Assume a conservative (for flight control applications) two second settling time. Five seconds after the simulations started, the system states and output have settled back to essentially zero.

Contrast the response of the above linear system to a general nonlinear system, such as the Block 40 flight control system. The nonlinear system is capable of achieving non-zero steady-state zero-input responses. For the above example, then there is no qualification that, after the doublet, the system states or output return to zero. In fact, for the Block 40 flight control system executing the test pitch doublet described, the states do not return to zero. This response is shown in Figure 4.21, where the maximum allowable command input is used to emphasize the problem. Note that the pitch rate does return to zero, but that θ does not. Also, while α may slowly converge on zero, the velocity, u, clearly diverges.

[4]The use of a doublet is motivated by the need to remain near the nominal trajectory to avoid violating the linear assumptions.

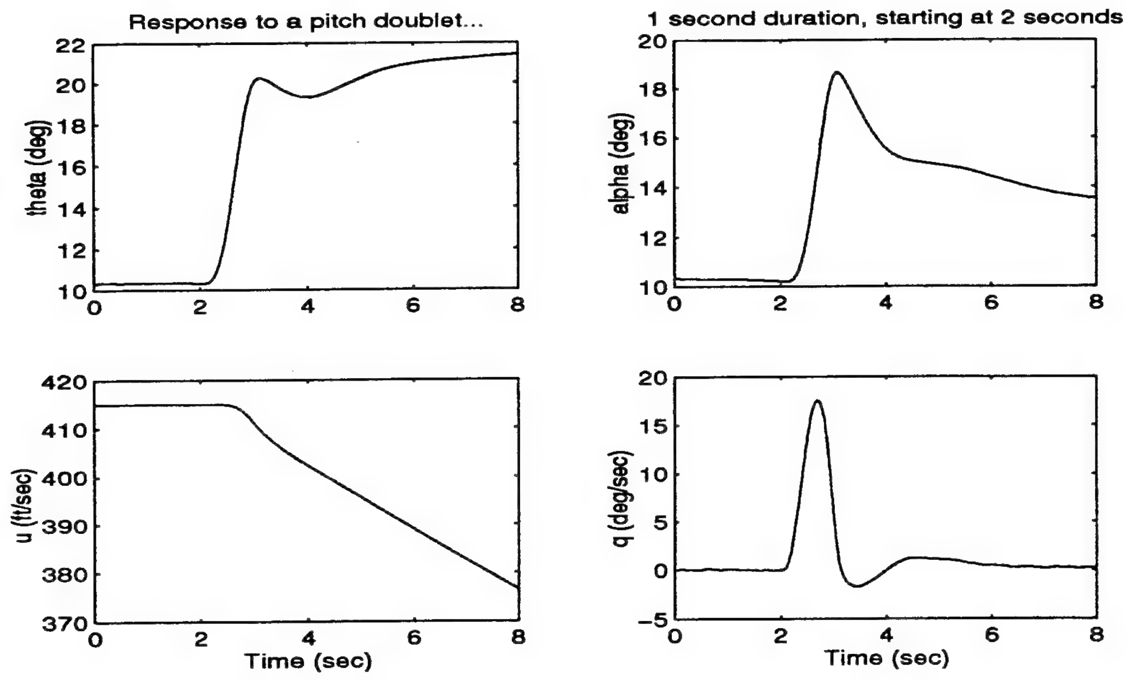


Figure 4.21 Nonlinear Response Characteristic of the Block 40 FCS

Despite the fact that a linear controller cannot match all of the performance characteristics of a nonlinear controller, for small amplitude inputs, even the LQG/PI controller with the numerical difficulties can be tuned to provide an acceptable match. The weighting matrices used to synthesize this LQG/PI controller are:

$$\mathbf{Y} = \begin{bmatrix} \left(\frac{1}{10.86}\right)^2 \times 7.5 & & \mathbf{0} \\ & \left(\frac{\pi}{324\times180}\right)^2 & \\ \mathbf{0} & & \left(\frac{\pi}{30\times180}\right)^2 \end{bmatrix} \tag{4.13}$$

$$\mathbf{U} = \frac{1}{25}\begin{bmatrix} \left(\frac{\pi}{19\times180}\right)^2 & & & & \\ & \left(\frac{\pi}{19\times180}\right)^2 & & \mathbf{0} & \\ & & \left(\frac{\pi}{21.5\times180}\right)^2 & & \\ & \mathbf{0} & & \left(\frac{\pi}{21.5\times180}\right)^2 & \\ & & & & \left(\frac{\pi}{30\times180}\right)^2 \end{bmatrix} \tag{4.14}$$

$$\mathbf{U}_R = \begin{bmatrix} \left(\frac{64\pi}{60\times180}\right)^2 & & & & \\ & \left(\frac{64\pi}{60\times180}\right)^2 & & \mathbf{0} & \\ & & \left(\frac{64\pi}{62\times180}\right)^2 & & \\ & \mathbf{0} & & \left(\frac{64\pi}{62\times180}\right)^2 & \\ & & & & \left(\frac{64\pi}{120\times180}\right)^2 \end{bmatrix} \tag{4.15}$$

Note that this is the same set of weights as that given in Equations (4.10)-(4.12), except for the rescaled $\mathbf{Y}_{11}$ term and the scalar multiplier on $\mathbf{U}$.

The test signals were selected as doublets in order to keep the aircraft within the bounds of the linear approximations for the duration of the eight-second simulations. The doublets are of a one second duration, starting at two seconds. For purposes of this research, the bounds were determined as the maximum perturbations for which an MMAE-based controller, using the Block 40 FCS and with a fully functional aircraft status, was able to maintain lock on the correct failure declaration. The magnitude of the doublets was selected to be 90% of the maximum control input, applied individually to each channel, for which this criterion was met. Note that column three of Table 4.6 gives the pilot commanded forces applied to the associated input given in column two. These forces are used as the magnitude of the input to the nonlinear simulation. Column five gives the equivalent command as calculated internally by the Block 40 FCS using software breakouts and command gradients [11]. These values are used as the magnitude of the input to the linear simulation and are required since the linear model does not include the nonlinear breakouts and command gradients.

Table 4.6 Test Doublets

Dynamic Channel	Associated Input	Command Magnitude	Commanded Variable	Converted Magnitude
Longitudinal	Pitch Stick	9 lb	a_n	1.64231 g's
Lateral	Roll Stick	2.7 lb	p	8.50 deg/sec
Directional	Pedals	54.9 lb	β	15.758 deg

Figures 4.22-4.24 display the results of applying the test signals to the LQG/PI controller test against the design model for a fully functional aircraft. For comparison, the response of the Block 40 FCS on the nonlinear truth model is also given. For the pitch doublet, only the longitudinal states are shown as the lateral/directional states are all essentially zero. Similarly, for the roll and yaw doublets only the lateral/directional states are shown. Although not exact, the state trajectories of the LQG/PI controllers follow the form of the Block 40's trajectories except in the lateral/directional responses to the roll and yaw doublets. In the response to the roll doublet, roll angle, roll rate, yaw rate all have slow initial responses in the opposite direction with respect to the Block 40's response. In addition, the actuators have moved in the opposite directions as well. By three seconds, however, all but the roll angle have moved back in phase, and even follow the form of the Block 40's response. Similarly, the states and actuators are 180° out of phase in the response to the yaw doublet as well. Applying a gain of -1 to the yaw input brings the response mostly back in phase, as shown in Figure 4.25, and this gain is added as a modification to the LQG/PI controller. Applying a similar -1 gain to the roll channel does not, however, achieve the same result, as shown in Figure 4.26, and is therefore not added to the LQG/PI controller.

4.8 Investigation of Possible Errors in the Linear Model

The fact that the trajectory moves 180° out of phase would seem to indicate a problem in the linear model, particularly in the input model since the actuators are out of phase as well. Moving the actuators in the wrong direction will certainly push the states in the wrong direction, so the actuators may well be the culprit. A review of the modified input matrix presented in Section 3.4.1 shows that the rudder, used for creating the yawing moments, effects only the lateral/directional states (the second four variables in the state vector defined in Table 3.1), while the flaperons, used primarily for rolling moments, effect both the lateral/directional and the longitudinal states. These facts may explain why reversing the sign largely corrects the phase problem for the yaw doublet

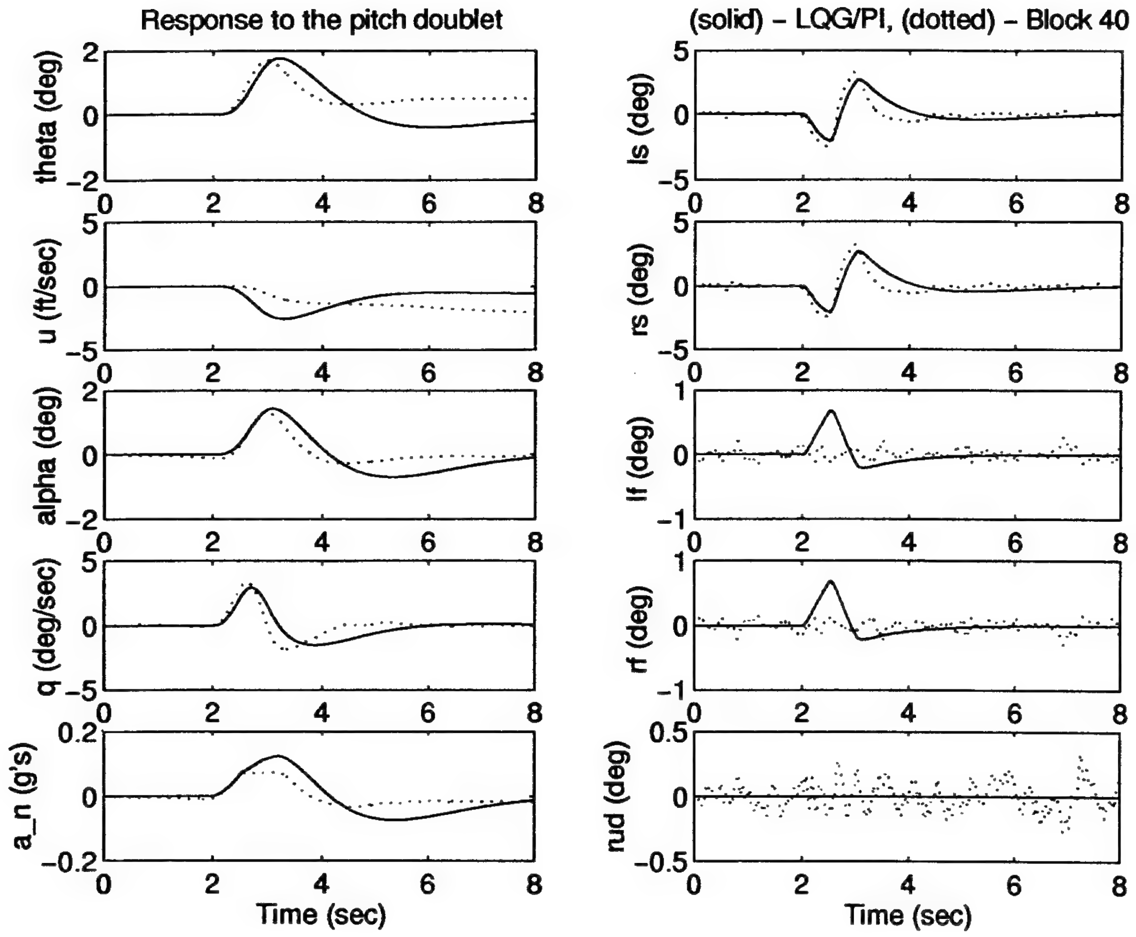


Figure 4.22 Linear Simulation of LQG/PI Controller - Pitch Doublet

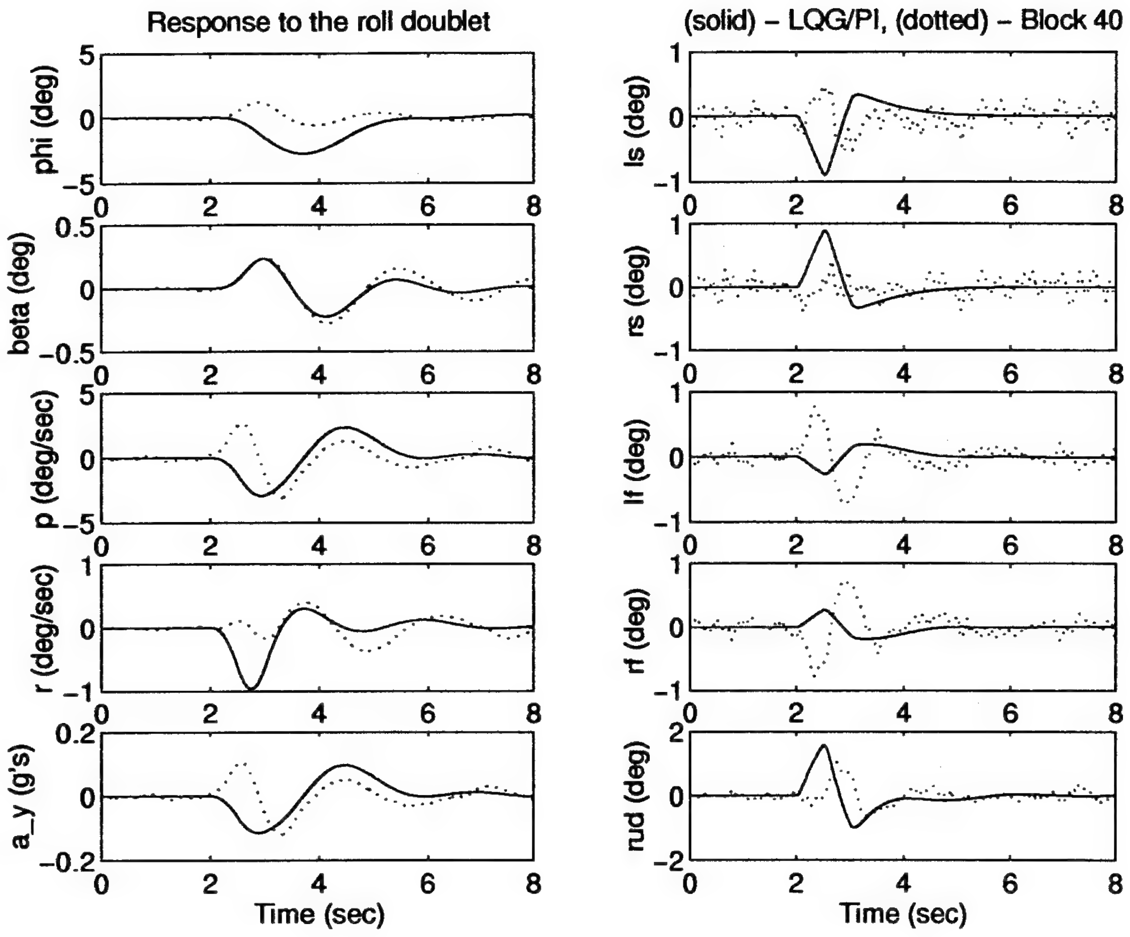


Figure 4.23 Linear Simulation of LQG/PI Controller - Roll Doublet

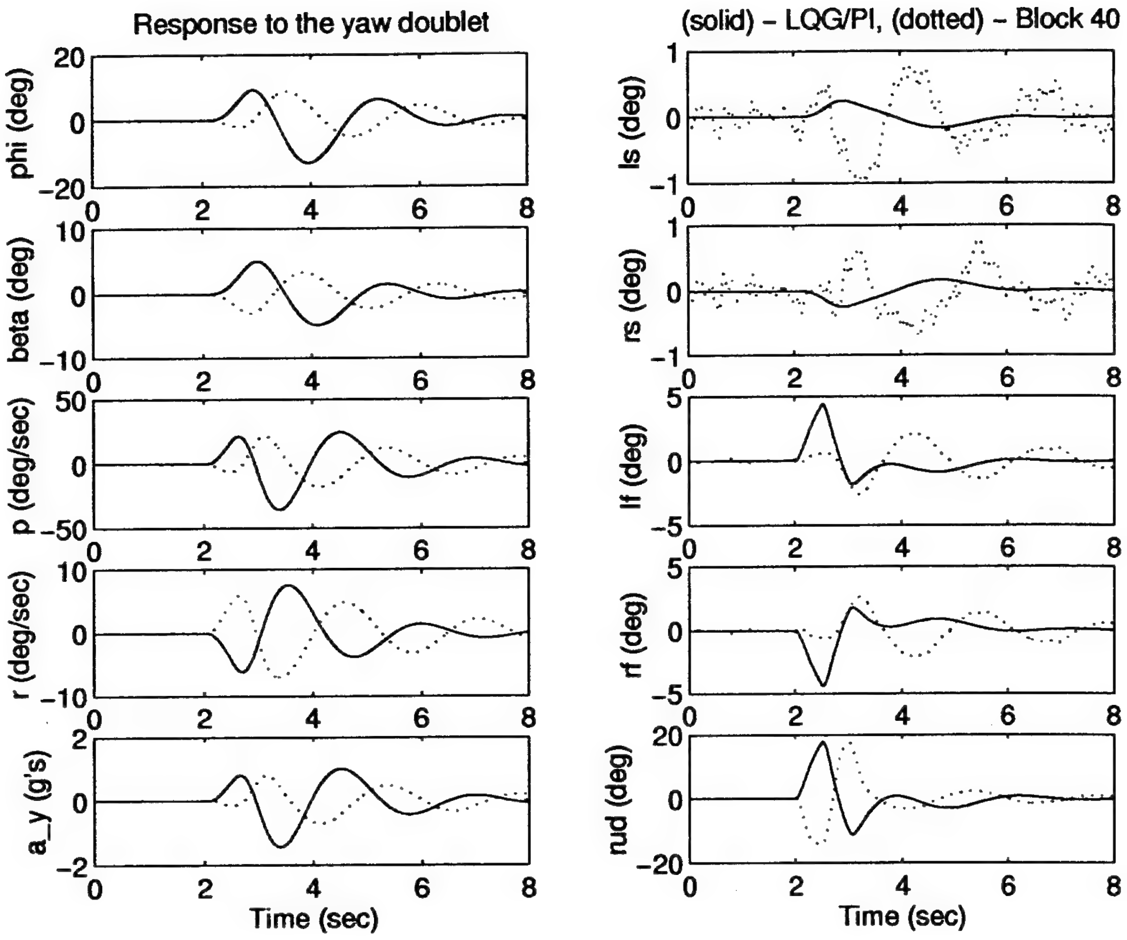


Figure 4.24 Linear Simulation of LQG/PI Controller - Yaw Doublet

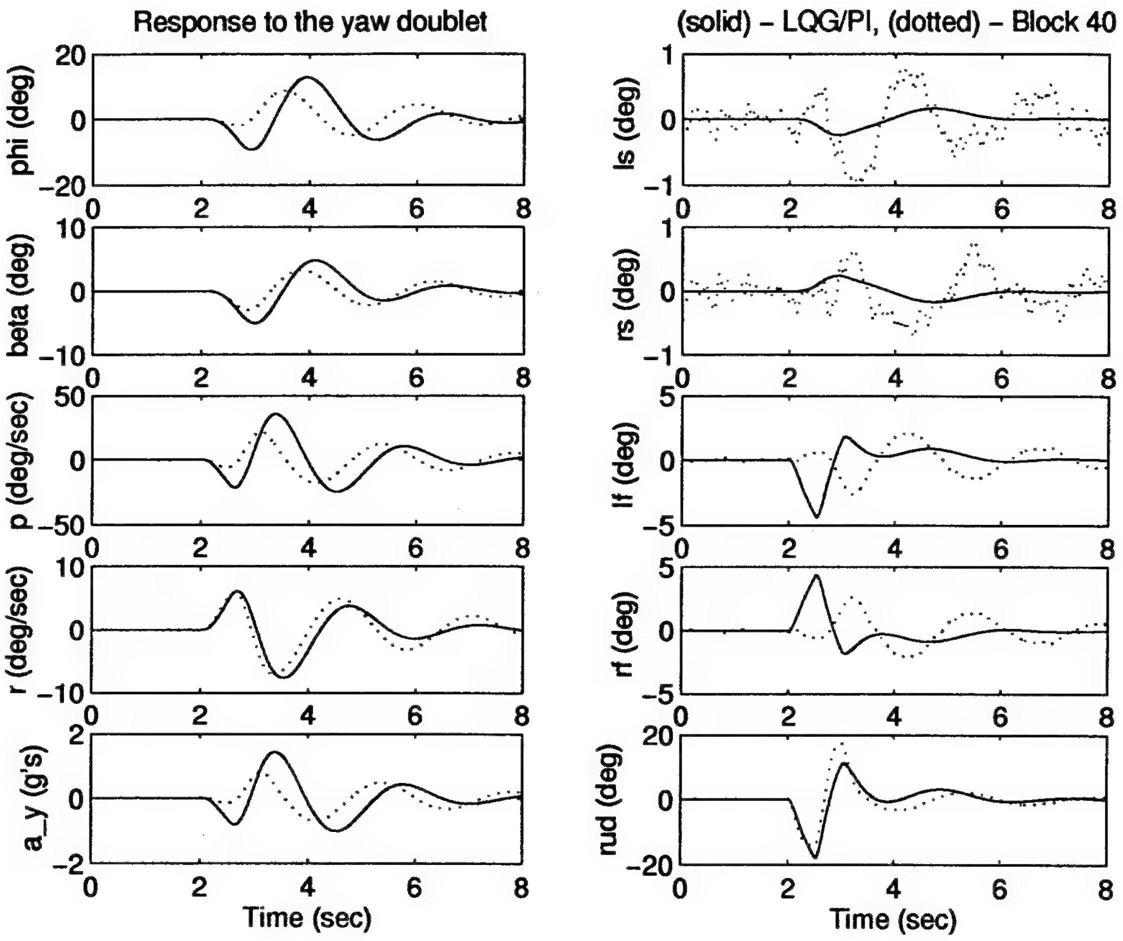


Figure 4.25 Linear Simulation of LQG/PI Controller - Negative Yaw Doublet

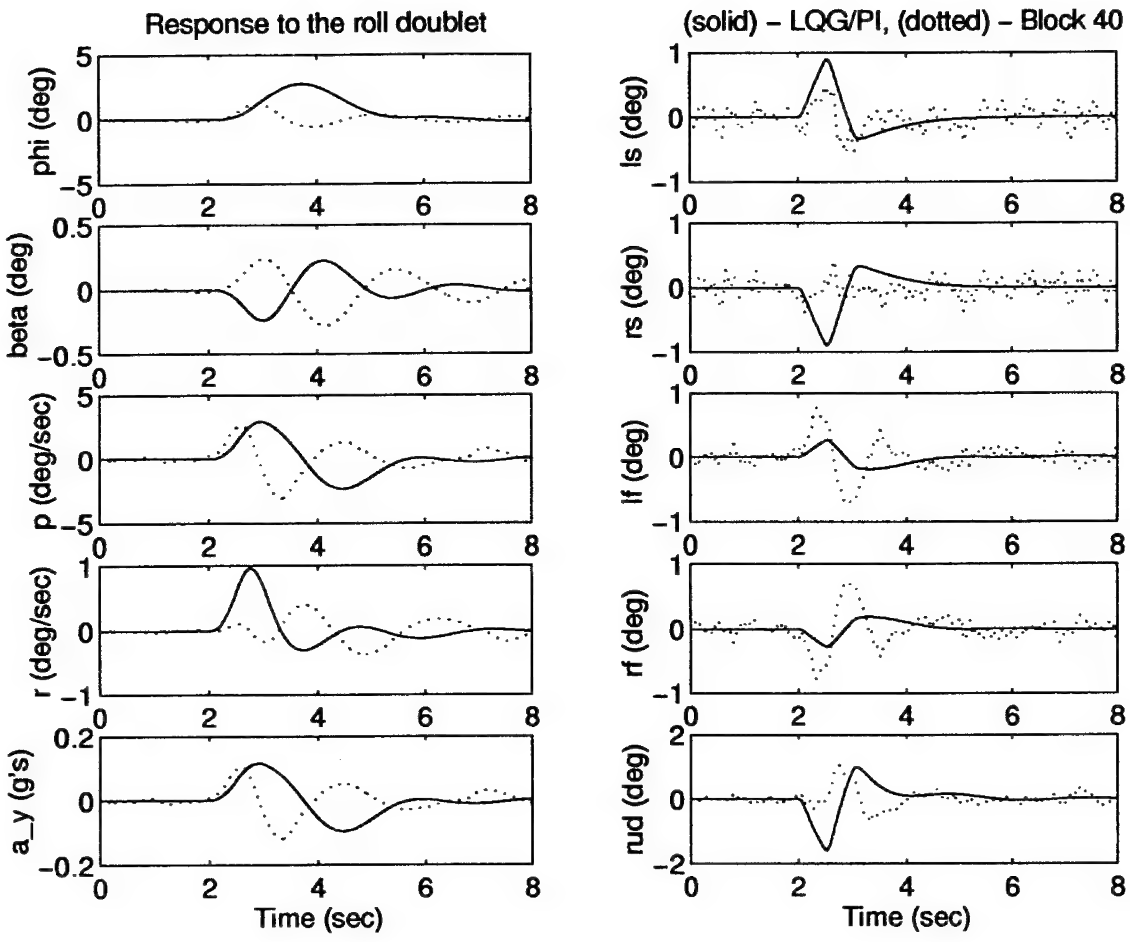


Figure 4.26 Linear Simulation of LQG/PI Controller - Negative Roll Doublet

but not for the roll doublet. Furthermore, the signs of the terms in the modified input matrix, $\mathbf{B}_{mod}$, are also informative. The modified input matrix, for the flight condition used in this research, is given here with the appropriate values for the stability derivatives substituted in:

$$\mathbf{B}_{mod} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1.0439 & 1.0439 & -0.2713 & -0.2713 & 0 \\ -0.0342 & -0.0342 & -0.0169 & -0.0169 & 0 \\ -1.8224 & -1.8224 & 0.1428 & 0.1428 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0069 & 0.0069 & -0.0003 & 0.0003 & 0.0170 \\ 4.4952 & -4.4952 & 6.2036 & -6.2036 & 2.8629 \\ 0.5173 & -0.5173 & 0.0654 & -0.0654 & -1.1652 \end{bmatrix} \tag{4.16}$$

Recalling the state and input vectors defined in Table 3.1, and based on the signs of the components of $\mathbf{B}_{mod}$, interpretations about the general response to an input can be made. Assuming for the stabilators and flaperons that an asymmetric command sends the positive input to the right surface and a negative input to the left, these interpretations are summarized in Table 4.7. Based on this assumption, which is supported by Control Surface Mixer block diagram for the VISTA F-16 [11], a positive input to the right stabilator produces the same response as a positive input to the right flaperon in the longitudinal channel, but different responses in the lateral/directional channel. Furthermore, the asymmetric inputs and the symmetric stabilator input have been verified

Table 4.7 Interpretation of the Modified Input Matrix

Positive Input	Primary Effect	Secondary Effect
Symmetric Stabilators	Nose up	Decrease in velocity
Asymmetric Stabilators	Roll left	Yaw left
Symmetric Flaperons	Nose down	Increase in velocity
Asymmetric Flaperons	Roll left	Yaw left
Rudder	Roll right	Yaw left

on the VISTA SRF simulation, seemingly isolating the problem to the terms of the input matrix corresponding to a symmetric flaperon or rudder input. Unfortunately, due to time restrictions, the exact location and source of the possible sign error remains unresolved.

Finally, with regards to the linear controller's performance, note that while this controller does give a rough match for the small test doublets used, it cannot in general match the nonlinear Block 40 FCS's performance. Specifically, the LQG/PI controller presented here does not meet handling qualities as defined by MIL-STD-1797A [36]. For example, neither the pitch response nor the roll response truly settles to a steady-state value. Also the controller is not capable of rolling the VISTA F-16 through 90° in one second as required. Despite these shortcomings, the implementation of the LQG/PI controllers into the MMAC may be insightful.

4.9 Implementation of the MMAC

Despite the fact that the LQG/PI controller was not able to meet the desired handling qualities, the LQG/PI controller bank was still put into the MMAC and tested against the eleven single failure conditions. The results, for which a representative summary plot is given in Figure 4.27, show that the MMAC falsely declared a right stabilator failure at 0.05 seconds, despite the fact that no failures were implemented until one second and the commanded doublet was not introduced until two seconds. The MMAC remains locked on to this declaration for the remainder of the simulation. There are several possible reasons for this false declaration. If the mismodeling described in the previous section does in fact exist, then the LQG controllers are likely supplying erroneous control laws. Also, the fact that the controllers are sub-standard due to the numerical problems may also contribute. Because of the poor LQG synthesis due to the numerics, the MMAC implementation was not pursued further, and the emphasis at this point is turned to the MMAE-based implementation using control redistribution.

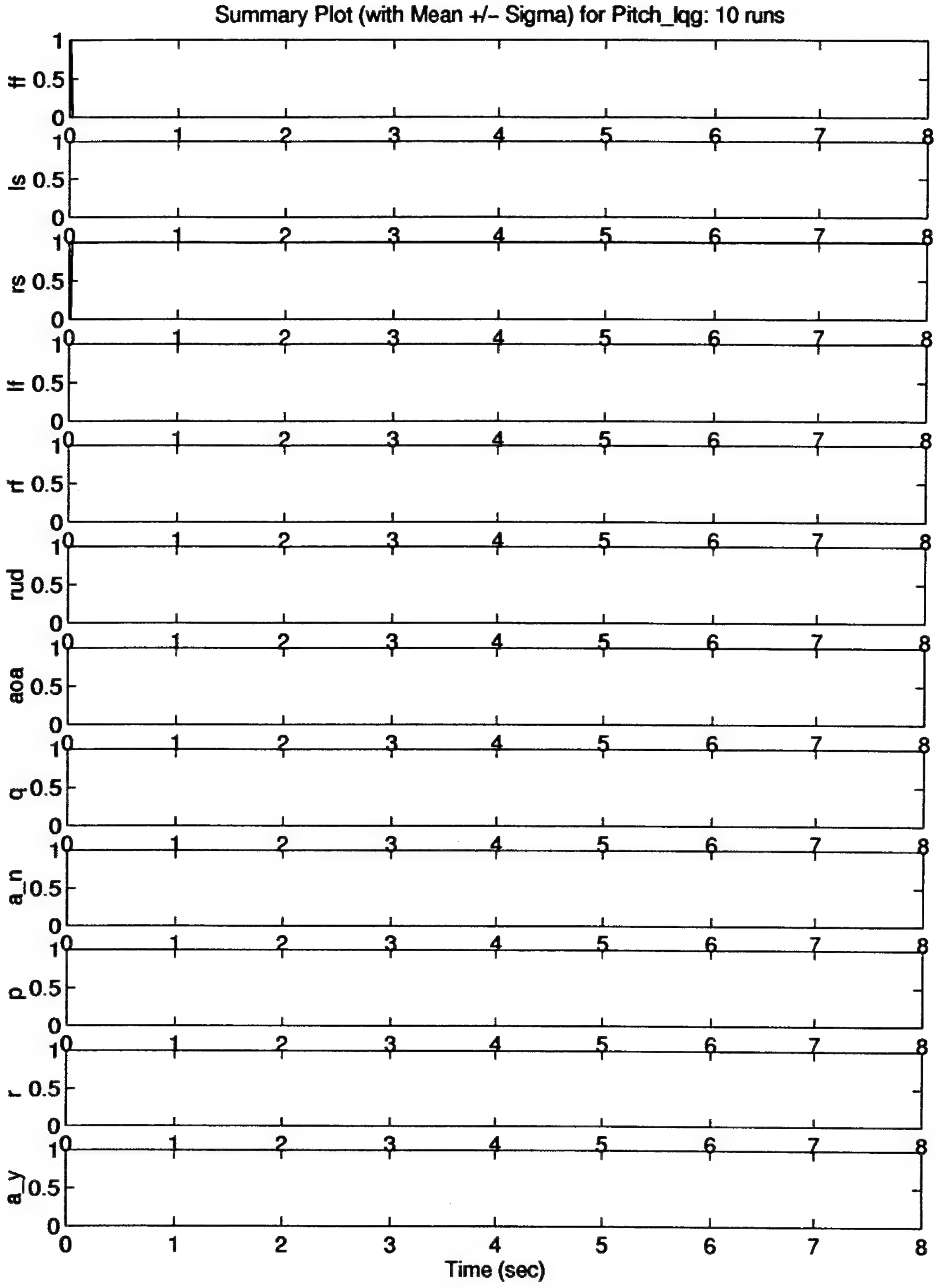


Figure 4.27 Summary Plot for the MMAC with LQG/PI Controllers

4.10 Control Redistribution

The method of control redistribution was tested using against two models: the linear design model developed in Section 3.4 and the nonlinear SRF VISTA simulation. The goal of the linear simulation is to verify the performance of control redistribution in an ideal environment, i. e. , where the design model used for both the redistribution and LQG/PI controllers is exactly the same as the truth model used for simulation. The nonlinear SRF VISTA simulation then incorporates a multiple model structure and demonstrates the ability of MMAE-based control redistribution to detect and compensate for sensor and actuator failures in a very realistic environment rather than an idealized one. The initial test doublets used are those presented in Table 4.6, having a period of one second and starting two seconds after the simulation has begun.

4.10.1 Verification of Control Redistribution. This verification is made using linear models and a completely deterministic system. The LQG/PI controllers, presented in Section 4.7, are allowed full-state feedback, and the control redistribution is artificially informed of the correct parameter realization. The failure is implemented at the start of the eight second simulation, and the trajectories of the aircraft with redistributed control are compared to the trajectory of a fully functional aircraft using only the LQG/PI controllers. The linear simulation incorporates both position and rate limiting of the actuators, though the rate limiting, which is not as critical due to the small magnitudes of the input commands, is not anticipated to be very accurate.[5] As saturations are induced in the actuators, and because pseudointegrators are used in the controllers, anti-windup compensation [24] is employed by imposing limits on the control inputs. Note that, as mentioned in Section 3.4, the VISTA F-16 with these LQG/PI controllers does not meet handling qualities (and is not even a satisfactory closed-loop controlled system). The intent of these simulations is only to verify the ability of control redistribution to match trajectories (regardless of controller), and the results are not necessarily indicative of desired performance.

[5]The rate is calculated based on the pseudorates, which are only a first order approximations to the true actuator rates.

As expected, the use of control redistribution on the linear simulation allowed the failed aircraft to follow virtually the exact same trajectory as the aircraft without failures when presented with the test doublets. So flawless was the performance using control redistribution, in fact, that rather than present the reader with repetitive figures, a composite test signal was created which exercised control authority in all three channels simultaneously. This composite test signal serves two purposes. First it poses a more challenging test signal (even inducing saturations due to the rate and position limits), and second, it reduces the number of plots presented, while still conveying the same amount of information. The composite signal is set up as an overlapping sequence of the same three pulses given in Table 4.6, except that each pulse is lengthened to have a three-second duration and the entire sequence of pulses begins one second after the simulation begins. Also, to avoid saturating the rudder on the fully functional aircraft used for comparison, the β input is reduced to ten degrees. Graphically the form of the input signal is given in Figure 4.28.

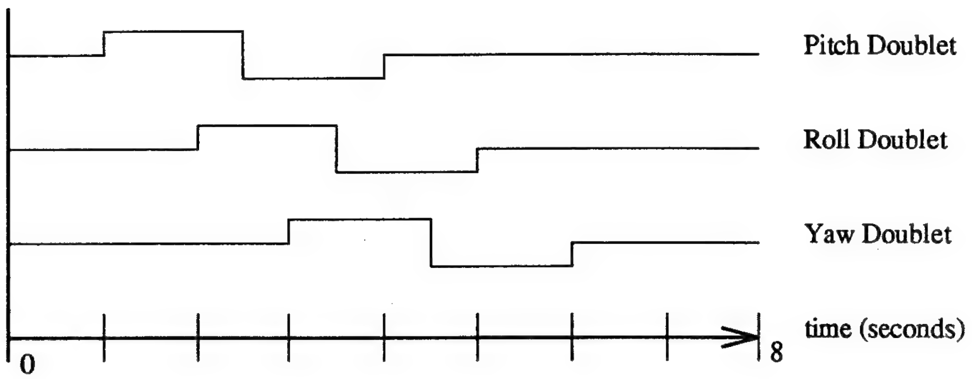


Figure 4.28 Composite Test Input

Figures 4.29-4.33 show the results of the linear simulations. Note that, for each of these figures, the first column is the set of longitudinal variables, the second column shows the set of lateral/directional variables, and the last column displays the control variables. Also, although not

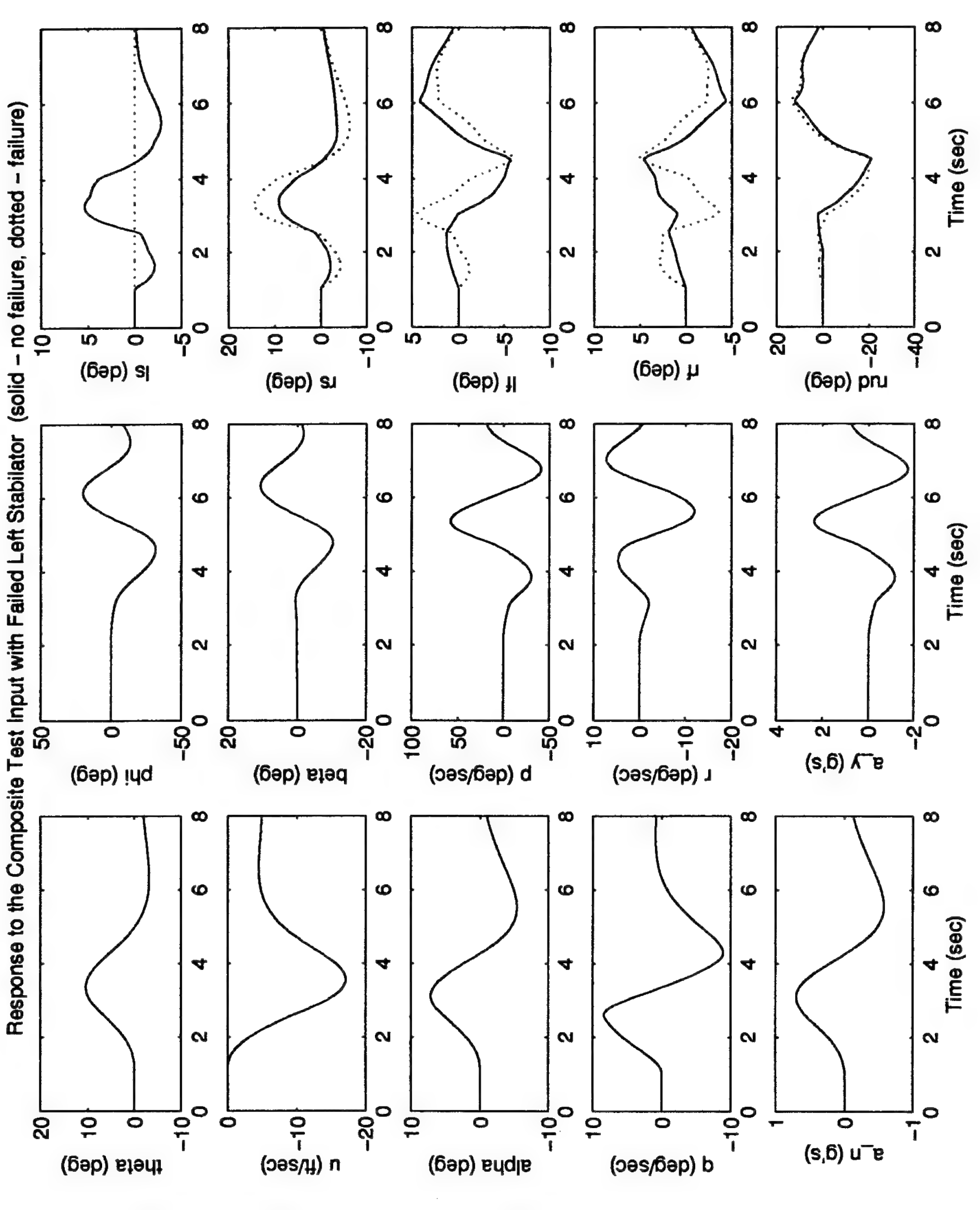


Figure 4.29 Verification of Control Redistribution - Left Stabilator Failure

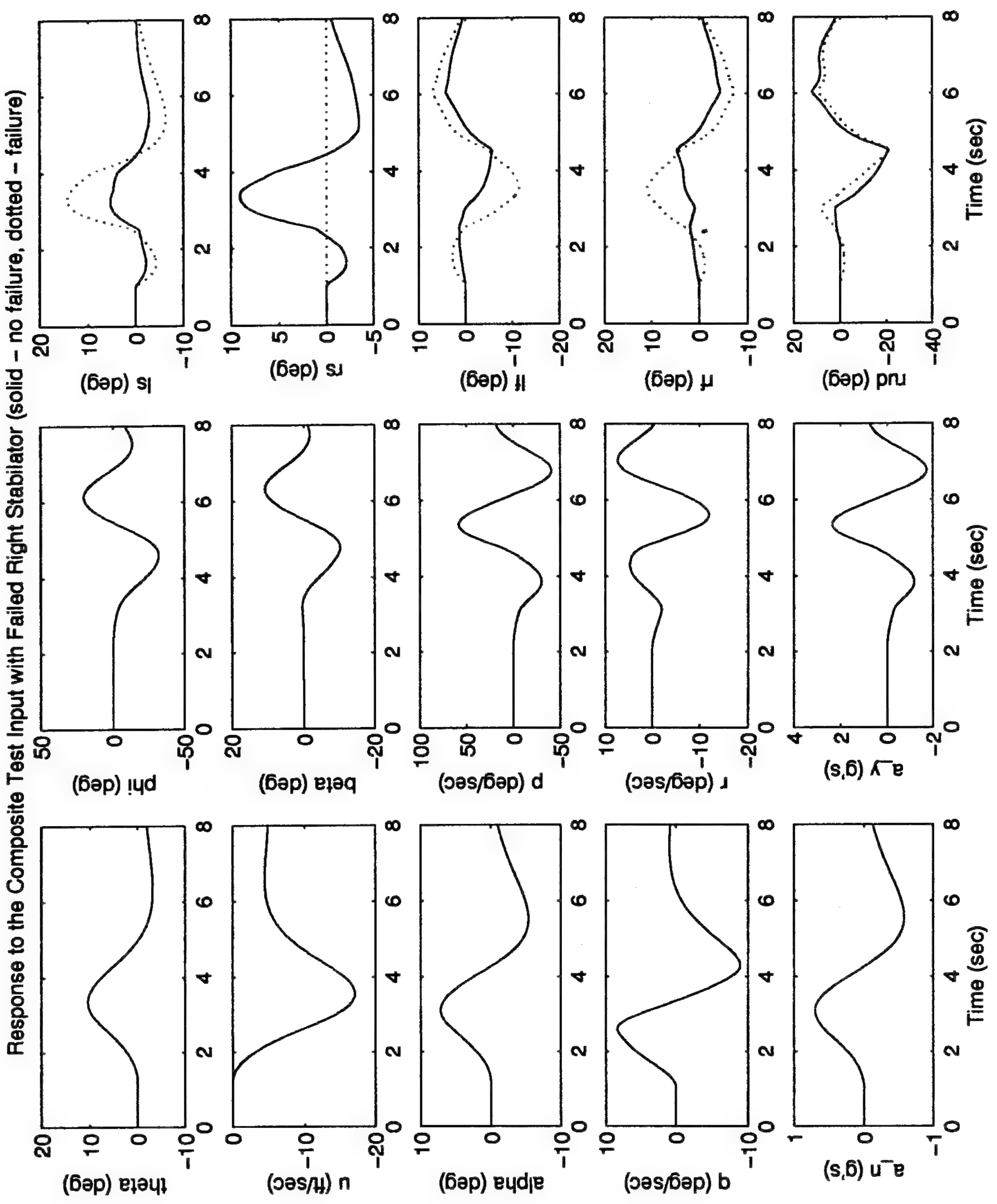


Figure 4.30 Verification of Control Redistribution - Right Stabilator Failure

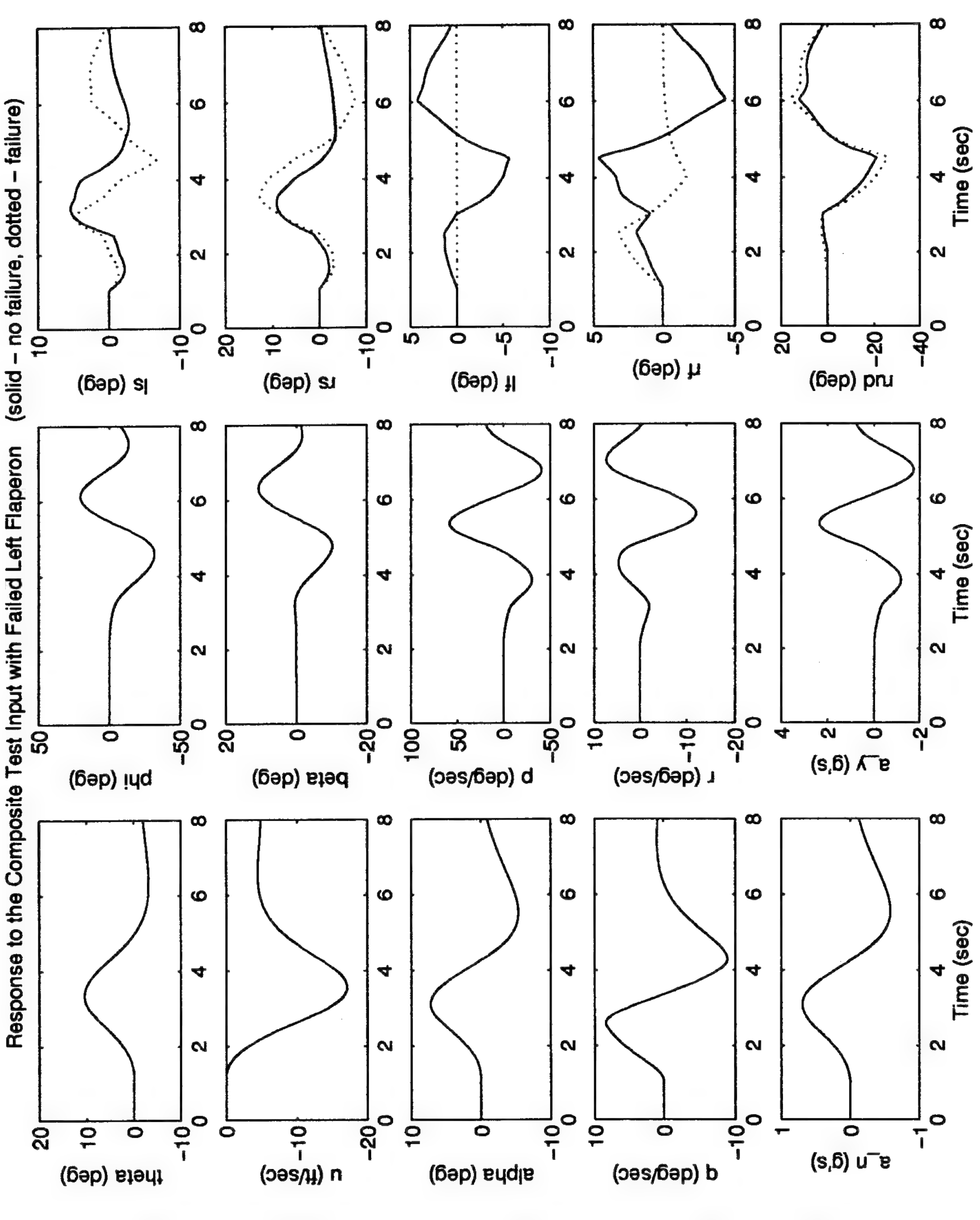


Figure 4.31 Verification of Control Redistribution - Left Flaperon Failure

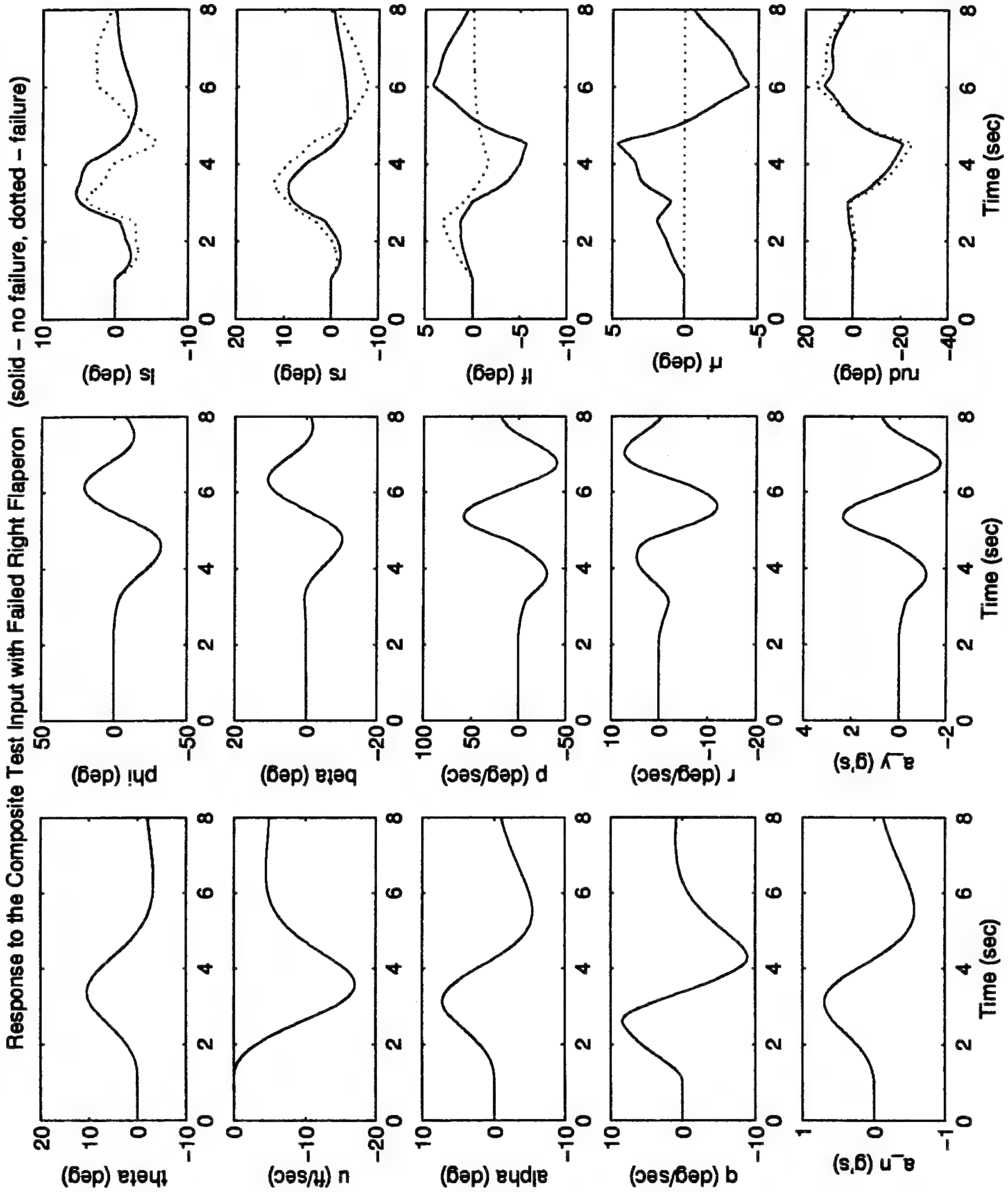


Figure 4.32 Verification of Control Redistribution - Right Flaperon Failure

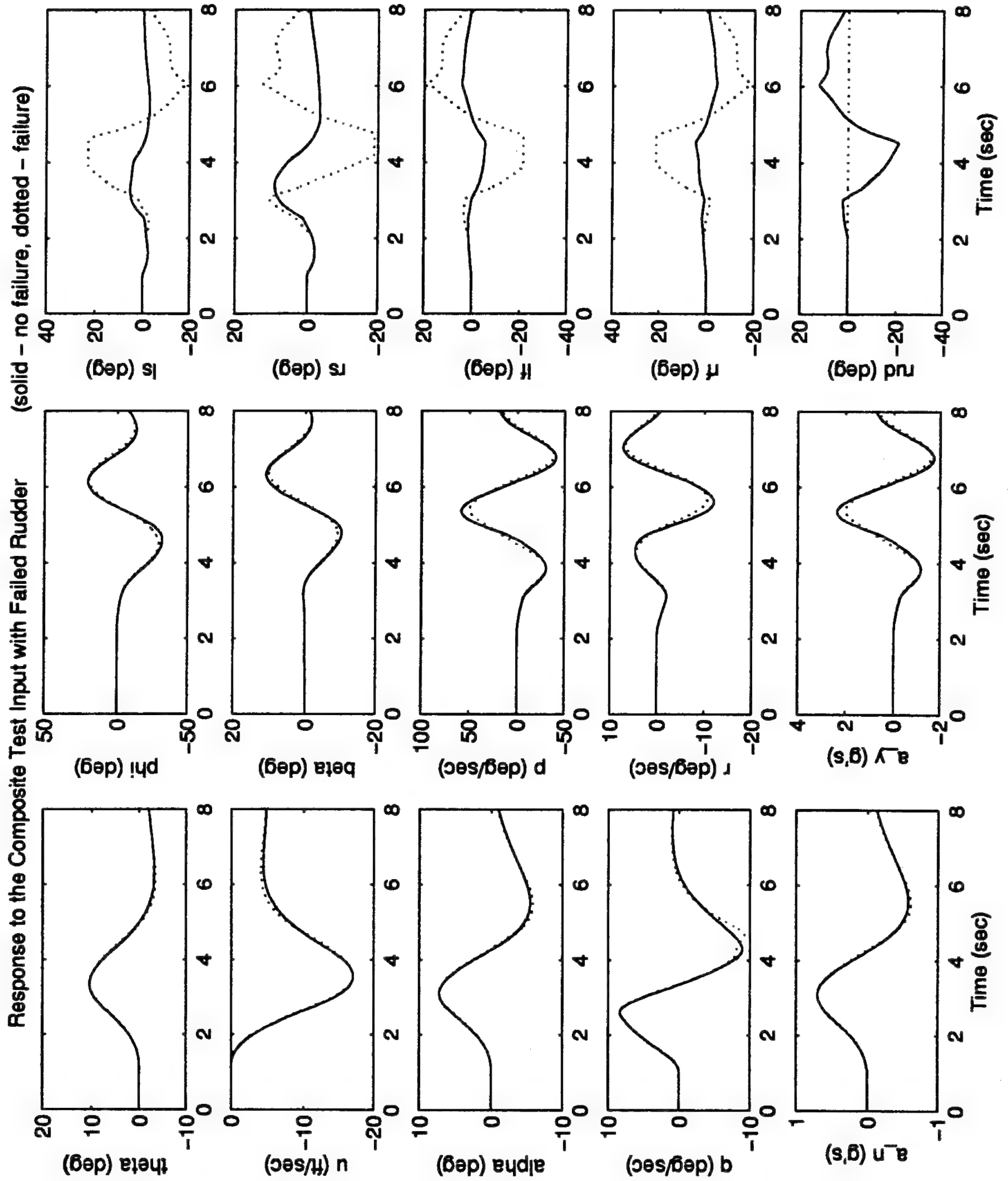


Figure 4.33 Verification of Control Redistribution - Rudder Failure

easily seen on most plots, each of these graphs really do contain two plots - a solid line corresponding to the response of the aircraft using control redistribution and a dotted line corresponding to the response of the fully functional aircraft without control redistribution. The control redistribution method proves to provide nearly identical performance with respect to the system states and outputs. Only on the actuator plots, where a difference is expected, can the two responses be easily differentiated.

Note that the flaperons and stabilators occur in pairs. While these sets are not truly redundant pairs, they do have some redundant properties that make it easier to cope with recovery from a stabilator or flaperon failure than recovery from a rudder failure. For example, in Figures 4.29-4.32, the magnitude of the remaining functional surfaces increases to reflect the additional control authority passed to them by the redistribution matrix, but the resultant positions are still reasonable. Contrast this to Figure 4.33, where a rudder failure causes the remaining surfaces to saturate under the additional load. However, because the controller is not attempting to drive the actuator hard into saturation, the state trajectory is still very close to that of the functional aircraft. The effects of position limiting on control redistribution can be enhanced by increasing the magnitude of the yaw doublet to 30 degrees. Figure 4.34 shows the resulting trajectories for the case of a failed rudder, where the redistributed controller is not able to follow the trajectory due to position limiting. Rate limiting is also a concern, but due to limitations of the linear model, which did not effectively implement rate limiting, the effect cannot be investigated here.

4.10.2 MMAE-based Control Redistribution. The evaluation of MMAE-based Control Redistribution is made using the nonlinear VISTA SRF simulation, with modifications described in Section 3.3, and incorporating the Block 40 FCS as the controller. The length of each simulation is set to be eight seconds, with a failure introduced one second after the start of the simulation. The test doublets are then introduced one second later. Note that a composite test signal such as that

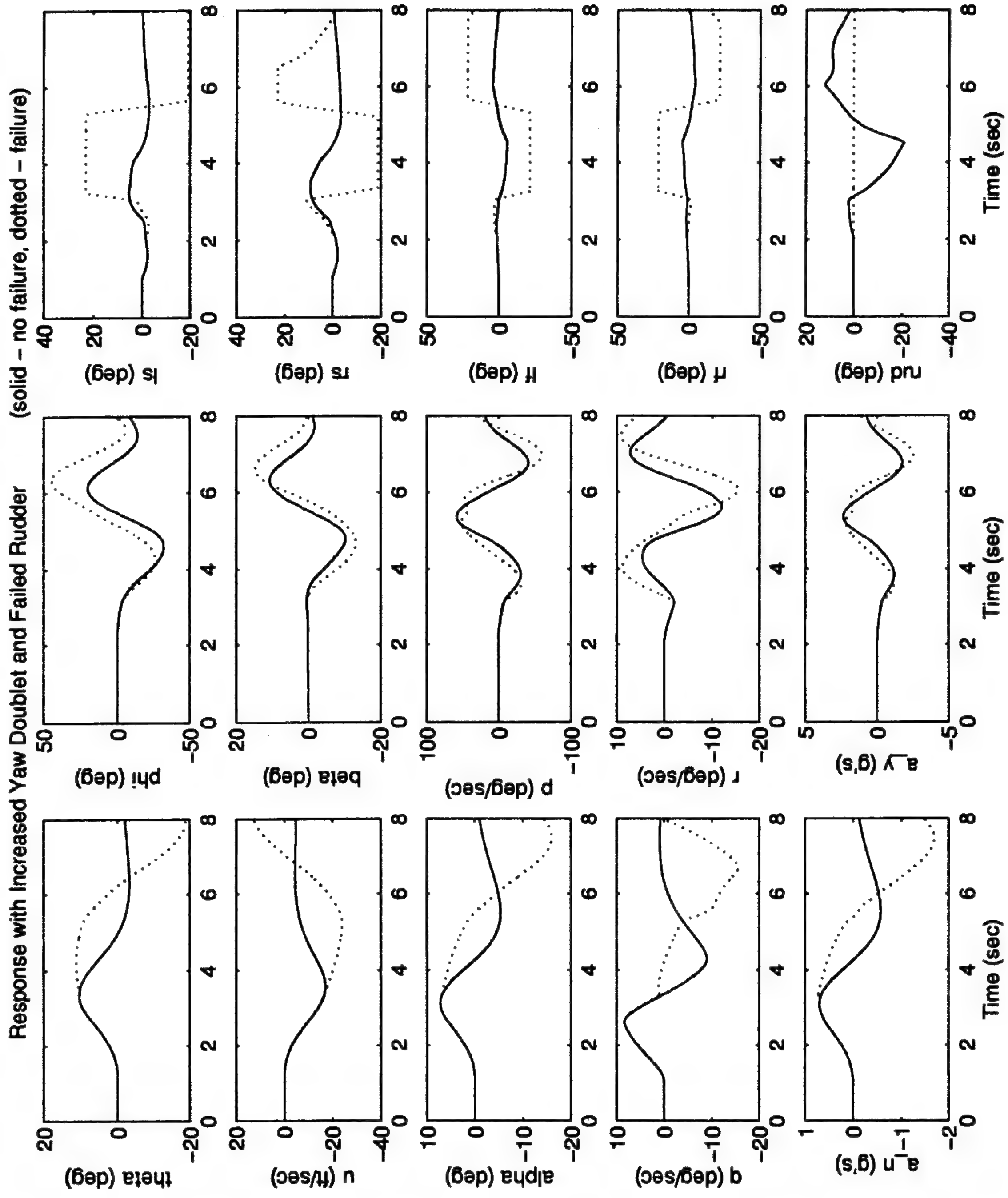


Figure 4.34 Verification of Control Redistribution - Effect of Position Limiting with Rudder Failure and Increased Yaw Doublet

used for the linear simulation was not possible on the nonlinear simulation without either leaving the self-imposed linear region, or else making the doublet amplitudes so small as to be uninteresting. Therefore, three separate test doublets, described in Table 4.6, are used here. Although the requirement to stay within the linear region of the MMAE may seem overly restrictive, Section 5.5 will present methods of implementation which will allow the operation of an MMAE-based controller to overcome this restriction.

Two sets of ten Monte Carlo runs are generated for each failure condition, and the results plotted together for comparison. The first set is made with the reference, fully functional aircraft and is used as a basis of comparison to evaluate the effectiveness of control redistribution. The mean (dashed line) and $\pm$ one standard deviation (dash-dotted line) are plotted. As the baseline, these same three lines will appear in each figure, though the standard deviation is so small that they may appear to the reader as one single line (the stochastic effects of the wind and noise is negligible compared to the magnitudes of the dither and test doublets). The second set incorporates MMAE-based control redistribution and implements the failure condition noted on the plot. Again, the mean (solid line) and $\pm$ one standard deviation (dotted line) are plotted, and in most cases these three lines will also appear as one solid line. Further, if the MMAE-based control redistribution method worked perfectly, then the two sets of mean$\pm$one standard deviations would overlap. However, an identical response is not anticipated considering the suspected mismatch, described in Section 4.8, between the nonlinear model, incorporated in the VISTA SRF, and the linear design model.

Although the original intent of this research was to use as the baseline a fully functional aircraft controlled only by the standard Block 40 FCS (driven directly by sensor outputs with no filter in the loop at all), a comparison of such a system and the same fully functional aircraft with an MMAE-based controller using the Block 40 flight control system revealed minor differences in the flight trajectories, as depicted in Figure 4.35 (the multiple traces are the two sets of mean$\pm$one

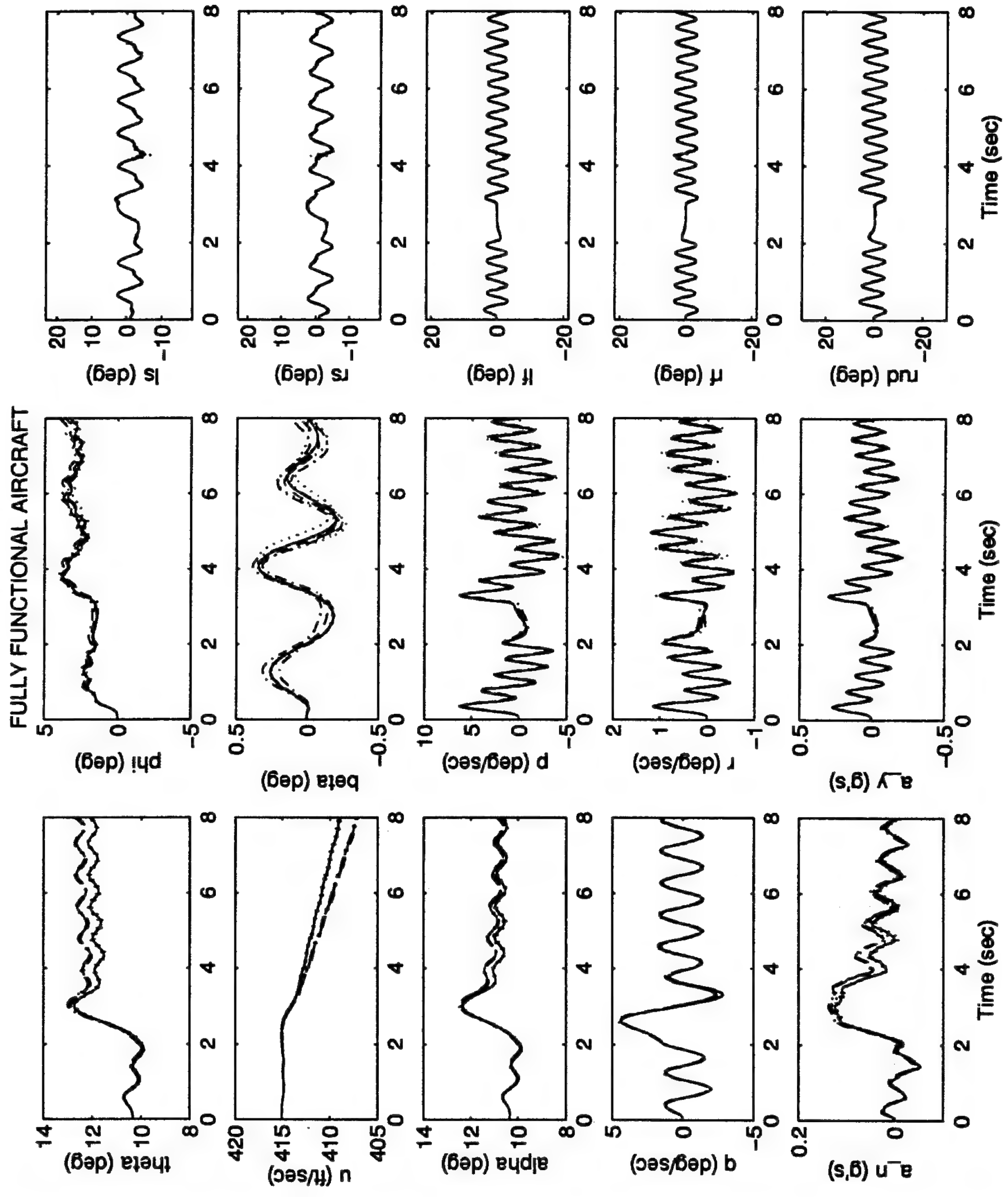


Figure 4.35 Comparison of the Original Block 40 FCS to an MMAE-based Controller using the Block 40 FCS

standard deviation trajectories for each aircraft). These differences arise from mismodeling in the linear models, and due to the fact that the aircraft is not really trimmed at a steady state condition, as noted in Section 3.4.1. Although not significant in terms of overall MMAC evaluation, these discrepancies will confound the analysis of the MMAE-based control redistribution method, as any deviations in state trajectory could be attributed either to the redistributed control or to the MMAE itself. Therefore, the baseline for comparison is defined as a fully functional aircraft controlled by an MMAE-based Block 40 FCS, rather than just the Block 40 FCS alone. This configuration is identical to that used in previous research [8].

Note that only flight conditions in which an actuator failure is experienced are considered, since sensor failures are not corrected by control redistribution. Sensor failures are compensated within the MMAE itself through the blending of the state estimates. Eide's MS Thesis [8] provides an analysis of an MMAE-based controller's response to sensor failures.

Figures 4.36-4.38 show typical comparisons between the (baseline) fully functional aircraft controlled with an MMAE-based Block 40 FCS and the aircraft experiencing an actuator failure controlled by MMAE-based control redistribution. The remaining comparisons are presented in Appendix A. Note that the same set of baseline mean±one standard deviation trajectory traces do appear in each figure, though they may seem different due to changes in scaling of the axis.

For stabilator and flaperon failures, the longitudinal states are able to track very well. There is a slight, steadily increasing, separation between the pitch angles, which results in an increasing flight path angle and therefore also appears as a slow separation in the velocity trajectory. Angle of attack and pitch rate are very closely matched by the redistributed control. Normal acceleration tracks with a slight offset bias. This positive bias may very well be the cause behind the slow changes indicated by the velocity and pitch angle. In the lateral/directional channel, the roll rate and lateral acceleration are seen to track very closely. A slight separation appears in the yaw rate,

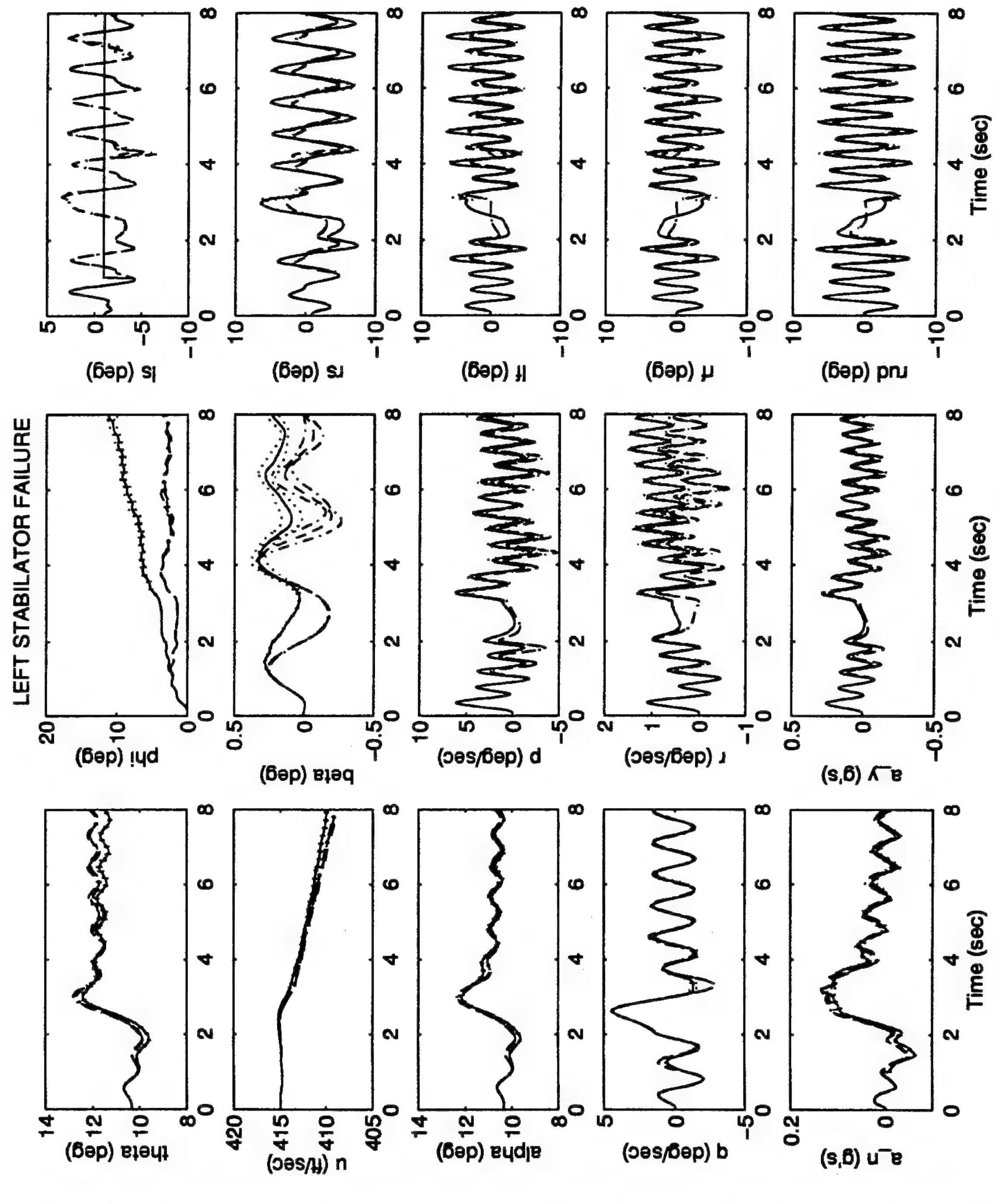


Figure 4.36 MMAE-based Control Redistribution - Pitch Doublet - Left Stabilator Failure

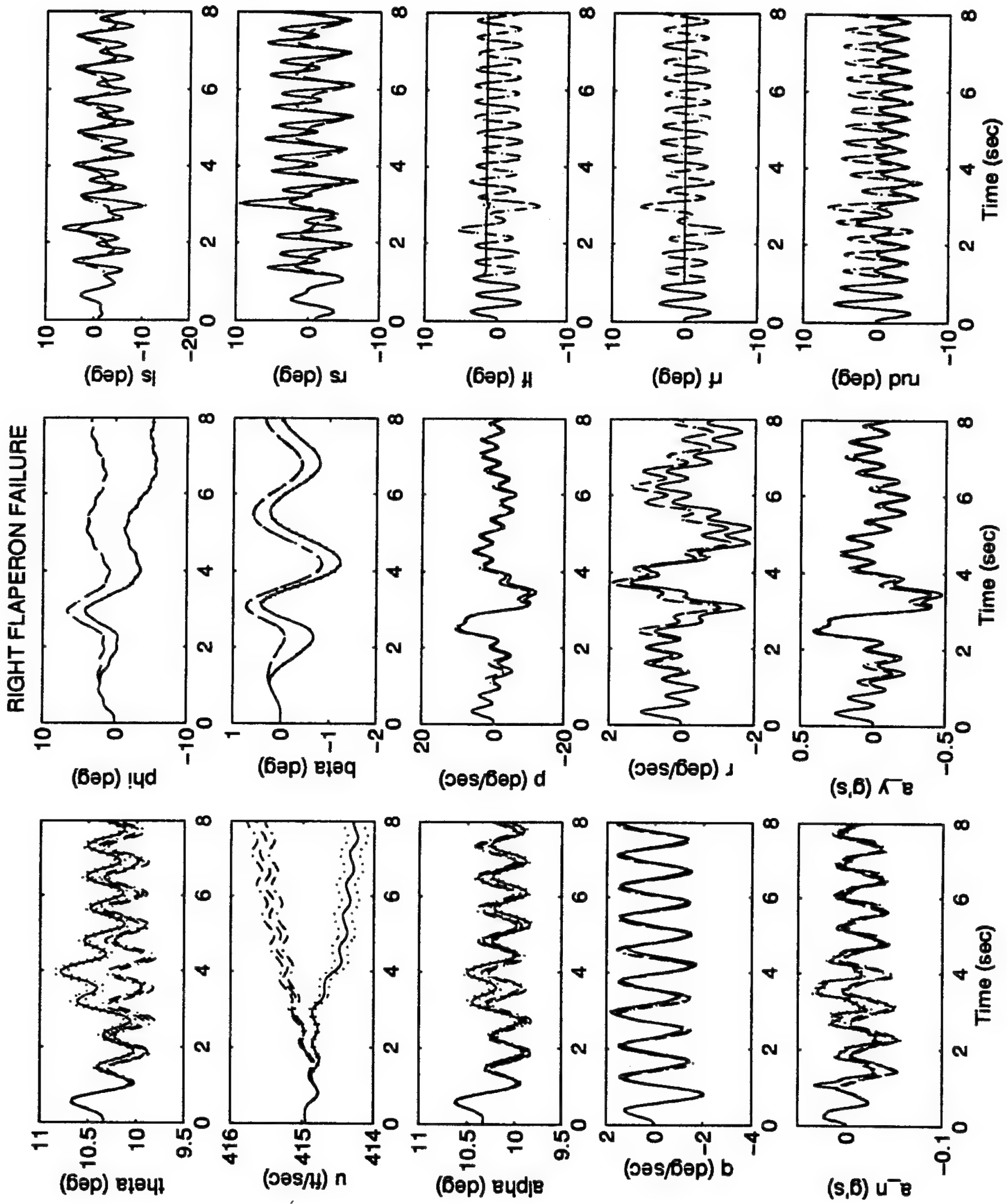


Figure 4.37 MMAE-based Control Redistribution - Roll Doublet - Right Flaperon Failure

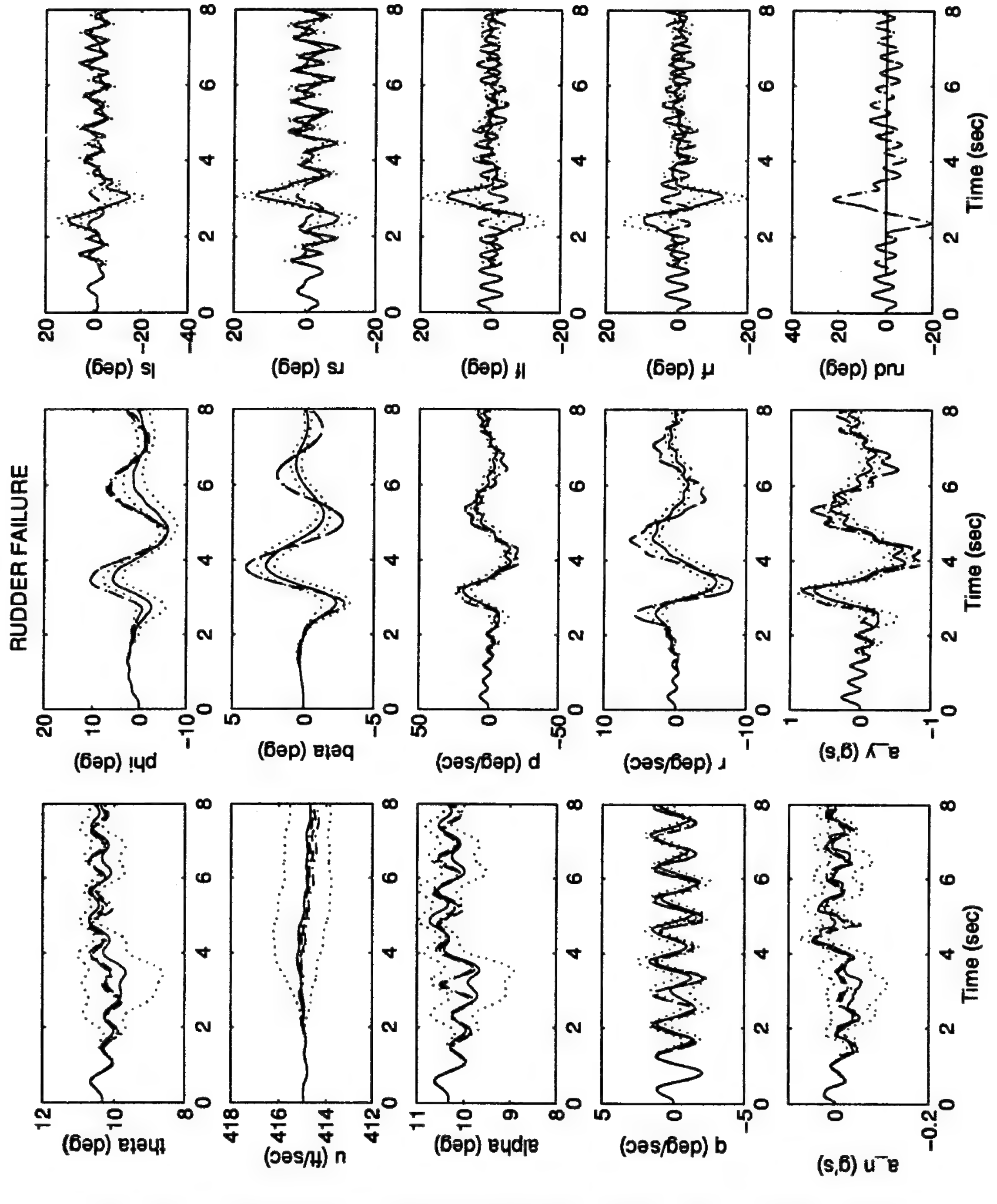


Figure 4.38 MMAE-based Control Redistribution - Yaw Doublet - Rudder Failure

as well as the sideslip and roll angles. Noting the scale on the graphs, the difference in sideslip angle can be dismissed as insignificant. Even the roll angle, which grows to a separation of approximately five degrees, and shows the worst separation of any variable, is of relatively little consequence since the pilot can manually adjust for such a small change in an angular component.

Based on the analysis of the linear simulation, the rudder failures are expected to present the most challenging scenario since there is the least amount of redundancy for this component. Figure 4.39 represents the worst response (the greatest amount of deviation, particularly with regard to the standard deviation traces), though the trajectories are still a relatively close match.

4.11 Chapter Summary

This chapter began by presenting unsuccessful attempts to reconcile the MMAC with the fact that the linear and nonlinear lateral acceleration models are grossly mismatched. Next, the dither signal was modified so as to maintain symmetry between the control surfaces, thereby greatly enhancing failure detection. Section 4.6 then detailed the numerical difficulties inherent in the discrete-time LQG/PI controller. Finally, Section 4.10 presented the performance of an MMAE-based controller using control redistribution with the Block 40 FCS. The next chapter will draw conclusion based on these results, and then provide recommendations for further research.

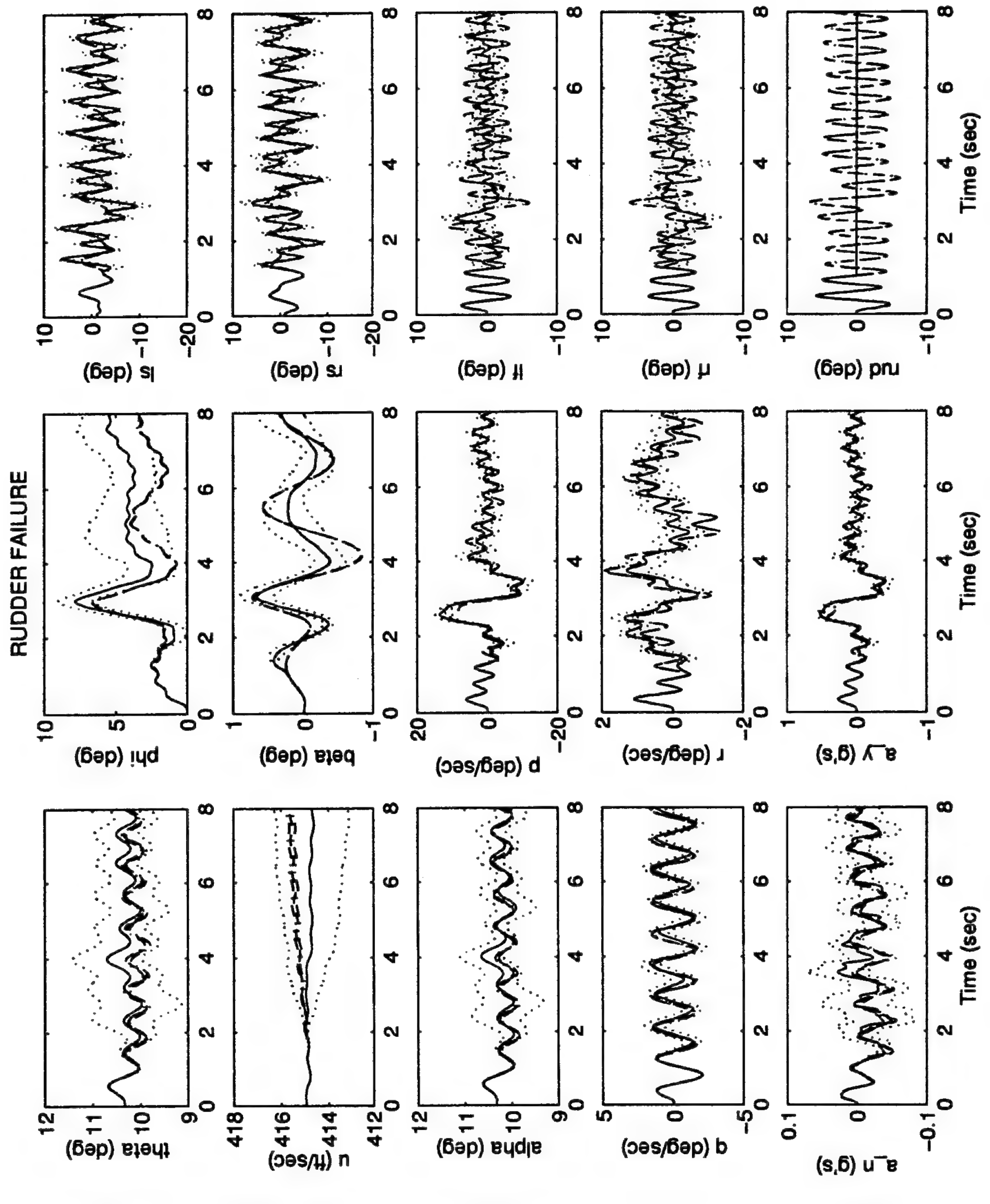


Figure 4.39 MMAE-based Control Redistribution - Roll Doublet - Rudder Failure

5. Conclusions and Recommendations

5.1 Chapter Overview

This chapter provides conclusions based on the results presented in Chapter 4. Section 5.2 draws conclusions based on the LQG/PI controller and its MMAC implementation. Concluding remarks concerning the MMAE-based control redistribution method are given in Section 5.3. Several enhancements to MMAE-based control redistribution are then presented for consideration in future research. These are followed by a closing description of a possible in-flight implementation of MMAE-based redistribution. Recommendations for further research are provided throughout this chapter where appropriate.

5.2 LQG/PI Performance

The numerical difficulties encountered in generating the $\mathbf{\Pi}$ matrix indicate that the LQG/PI synthesis of a controller using an output linearly dependent on the state derivatives will be problematic. These problems may possibly be circumvented by "corrupting" the linear output model with erroneous numbers just big enough to remove the linear dependencies, but small enough to maintain acceptable generation of the control variable [44]. For example, a pitch rate output might be constructed from $\dot{\theta} + \epsilon\alpha$ instead of just $\dot{\theta}$, where ϵ is a suitably small number. Initial attempts at this technique, however, were unsuccessful, as the required magnitude of ϵ to alleviate the linear dependency on state derivatives was found to be so large that the integrity of the output model was unacceptably degraded. Similar methods of such intentional "corruption" may prove more fruitful, though the use of such ad hoc methods may not yield a procedure readily portable to other applications. Other possible solutions include using different design models, such as the short period model for the longitudinal controller, or possibly using a balanced realization to improve the conditioning of the matrices. Note that, for designs based on the F-16, which is statically unstable, an inner loop must be used to first stabilize the system before the balanced method can be used.

Finally, different output variables may be selected which are not dependent on the state derivatives, though this precludes the use of C^*, q, and a_n, all desirable control variables for the longitudinal channel.

Because of the unsuitable controllers generated by the LQG/PI synthesis, the MMAC implementation was not successful and yielded few insights. These results should not, however, dissuade further research into the use of multiple model adaptive control. With a properly functioning controller bank, the MMAC method has great potential for detecting and compensating for sensor and actuator failures.

5.3 Control Redistribution Performance

The MMAE-based control redistribution has shown excellent results and yields a design which, because it augments instead of replaces the existing flight control system, is easily implemented for flight testing. The MMAE front-end is capable of detecting most single failures in less that 0.5 second. The hardest to detect failures are the angle of attack and the yaw rate sensors, but both are still detected in under one second. Because these two slowest declarations involve sensor failures, scalar residual monitoring could be used to reduce this detection time, if desired.

5.3.1 Partial and Multiple Failures. Although partial failures and multiple failures remain to be tested using a full implementation of MMAE-based control redistribution, Eide [8] did investigate the ability of the MMAE to detect (but not control) partial and multiple failures. Since an MMAE can completely compensate for sensor failures,[1] what remains to be considered is the ability of control redistribution to account for partial and multiple actuator failures. Because control redistribution is able to compensate nearly perfectly for actuator failures, a partial failure may best be labelled as a complete failure. This conservative declaration results in a more severe

[1]Note that Section 5.6 will propose an alternate control method in which the redistribution technique is applied to measurements as well.

reduction in control authority than may be truly necessary, but it also maintains the number of elemental models required in the MMAE at one per actuator. For critical components, additional models hypothesizing partial failures may be included in the discretized parameter space. Multiple failures are readily admitted through the use of a hierarchical structure. Investigations of partial and multiple failures are strongly recommended for future research.

5.3.2 Modeling. Certainly the performance of any control scheme hinges on the accuracy of the design model in representing the real world (or truth model). Some doubts as to the relationship between the design model and truth model, as well as to the fidelity of the SRF VISTA simulation itself, have been raised in this research. A more thorough analysis may yield better models (and correspondingly better performance), particularly with respect to the input matrix, **B**, which is critical to the synthesis of the redistribution matrices. At the same time it must be noted that control redistribution is not limited to a time-invariant model. By simply relinearizing at every sample period to enable the redistribution synthesis, a high order, nonlinear model can also be used. Note that relinearization of the control redistribution model in no way effects the control law generated by the Block 40 FCS; it only enhances the accuracy of the redistribution.

5.3.3 Preventing Saturations. The use of control redistribution resulted in performance by a failed aircraft nearly identical to that of the fully functional aircraft. In fact, when simulated on the linear truth model, where the design model matches the truth model exactly, the two trajectories were indistinguishable, provided no saturations occurred. If desired, the increased likelihood of saturations, due to the additional load of the redistributed control, can be prevented by attenuating the input commands. However, some loss of performance is expected for a failed aircraft, and control redistribution has the desirable trait that it only shows this degradation at the limits of performance.

Note that the range in which redistributed control can completely compensate for a failed aircraft (before reaching saturations) is largely defined by the redundancy inherent in the control surfaces. For example, while the stabilators and flaperons occur in matched pairs, the VISTA F-16 has only a single vertical tail and therefore control redistribution must work hardest to maintain commanded performance when experiencing a rudder failure. This fact is explicitly shown in the packed redistribution matrix, $\mathbf{D}_a$, described in Section 3.8 and presented here for reference:

$$\mathbf{D}_a = \begin{bmatrix} 0 & 1.0000 & 1.1037 & -1.1037 & -1.2719 \\ 1.0000 & 0 & -1.1037 & 1.1037 & 1.2719 \\ 0.9060 & -0.9060 & 0 & 1.0000 & 1.1524 \\ -0.9060 & 0.9060 & 1.0000 & 0 & -1.1524 \\ -0.7862 & 0.7862 & 0.8678 & -0.8678 & 0 \end{bmatrix} \tag{5.1}$$

where each column gives the redistribution for the corresponding actuator as listed in Table 3.3. Note that the fifth column, corresponding to the redistribution of the rudder, contains the largest magnitudes, representing the largest redistributed commands being sent to each of the remaining actuators. This is corroborated by the fact that saturations occurred only for rudder failures in Section 4.10.1. An aircraft with multiple rudders (vertical tails), such as the F-15 or F-22, would alleviate this problem and increase the range for redistributed control (before hitting saturations).

5.3.4 Intermittent Failures. The detection of intermittent failures by the MMAE must be treated somewhat differently than it has in the past, because control redistribution allows a failed aircraft to respond in a manner indistinguishable to that of a fully functional aircraft. For example, suppose that the right stabilator fails. The aircraft trajectory begins to deviate from that expected by the MMAE, and within a fraction of a second the correct failure condition is declared. Suppose that the failure is intermittent and that five seconds later the right stabilator becomes fully functional again. Because, for moderate inputs, as described in Section 5.3.3, the performance of the failed aircraft matches that of the fully functional aircraft, the MMAE has no reliable way of distinguishing between the fully functional hypothesis and the previously declared failure hypothesis. The MMAE is therefore unable to reverse its decision and decide that an actuator in fact either has not failed (a false alarm) or is not currently failed (an intermittent failure). Fortunately, there is little loss of performance due to the accuracy of control redistribution. Even so, the MMAE can be forced to search for intermittent failures by periodically, or at pilot-specified times, resetting the failure hypothesis back to that of a fully functional aircraft and setting the redistribution matrix back to an identity matrix. In the third robustness technique recommended in Section 5.5, the MMAE essentially accomplishes this restart every time it re-engages by passing through the nearly nominal region.

5.4 Dither

Although the modified dither signal, as presented in Section 4.5, appears to result in symmetric commands sent to the actuators, the final tuning summary plot shown in Figure 4.11 reveals that the left stabilator and flaperon failures are still detected more slowly than those of the right stabilator and flaperon, indicating room for more improvement in the dither signal. Because the ability of the MMAE to detect failures relies on excitation from the dither signal at benign flight conditions, a better dither might in fact improve the detection times for all failures, though only at benign flight conditions (dither is disabled when the pilot commands an input through the force stick or pedals).

5.5 Enhancing Robustness of the Linear MMAE

The effectiveness of MMAE-based control redistribution relies on the ability of the MMAE to declare the correct failure status of the aircraft accurately. The MMAE used in this research accomplishes this detection of failures using linear Kalman filters. An obvious pitfall then is the effect on the system when the aircraft moves away from the linear design point. Eide [8] investigated this effect and noted that the MMAE quickly failed as the aircraft moved off nominal, i. e. , the MMAE began to give completely erroneous failure declarations. While the use of gain scheduling can compensate for flight patterns involving varying dynamic pressure (such as those caused by changing altitude or speed), changes in attitude (such as flying with wings not level) still upset even the scheduled MMAE. For use in a highly maneuverable fighter aircraft, then, the robustness of the MMAE must be enhanced to account for flying off the nominal design point.

Three methods are suggested for such an enhanced MMAE implementation. The first uses extensive gain scheduling to account for all possible flight conditions and aircraft orientations. For example, a multidimensional lookup table based on dynamic pressure, roll angle, and sideslip angle (and possibly other variables) might prove successful. The complexity and magnitude of such a table, however, deems it not practical for a real implementation for the entire flight regime. Additionally, the variables needed to navigate the table are those estimated by the MMAE, which in turn relies on the scheduled data from the table, resulting in an undesirable interdependence. More pertinent to the current design, the issue of estimating the lookup table's independent variable raises the question of whether or not the MMAE should provide a dynamic pressure estimate since both the Block 40 FCS and the LQG controller presented in this thesis rely on pressure measurements to schedule variables. Adding a dynamic pressure estimate to the MMAE is recommended for future research.

The second method is perhaps more feasible, but just as complex, and involves using higher order models within the MMAE's Kalman filters. For example, the use of nonlinear filters would increase the robustness of the MMAE. Whether or not an MMAE using nonlinear filters is capable of withstanding the extreme maneuvers required in a aerial dog-fight remains to be demonstrated. Because of the additional computational burden required for the implementation of nonlinear filters, however, this method may not be as suitable as the third technique.

The third technique, and the method recommended for future research, is simply to admit that the MMAE is not effective at off-nominal flight conditions, and so use additional logic to disengage the MMAE except when the aircraft is determined to be flying in a nearly nominal condition, nearly nominal being defined as being in approximately steady-state, trimmed flight. This decision might be based on a comparison of state variables such as angular rates, sideslip angle and roll angle to a predetermined threshold. Only if all variables fall below the specified threshold is the MMAE engaged. Alternatively, the MMAE could be engaged manually by the pilot if a failure is suspected and the aircraft is purposely brought to a nearly nominal flight condition.

This third technique still relies on gain scheduling to be effective at different dynamic pressures, but this is not a considerable modifications since gain scheduling already occurs within the Block 40 FCS. In this implementation, the controlling logic must also ensure that the parameter estimate, $\hat{\mathbf{a}}$, is buffered when the MMAE is disengaged, so that the control redistribution back-end maintains the failure declaration until the MMAE is again engaged and allowed to redeclare the failures as appropriate. Although this configuration only enables the MMAE during nearly nominal flight, because the MMAE requires less than a second to declare the correct failure, the aircraft actually only needs to pass briefly through the nearly nominal region. Also, in such an implementation where the aircraft is generally not in a benign flight condition, the need for dither is eliminated completely.

5.6 Redistribution Applied to Sensor Failures

One problem with the suggested technique for robustness enhancement is that sensor failures are compensated via the state estimates, $\hat{\mathbf{z}}$, which are generated by the MMAE. With the recommended method described in Section 5.5, where the MMAE is disengaged when the aircraft is off-nominal, this ability to compensate for sensor failures is lost. A solution may be to apply the redistribution technique to the measurements as well as to the controls while the MMAE is disengaged. The development of measurement redistribution follows that presented in Section 3.8 except that the defining relation is now:

$$\mathbf{z} \approx \mathbf{z}_r \tag{5.2}$$

or

$$\mathbf{H}_{aug}\mathbf{x}_{aug} \approx \mathbf{D}_{si}\mathbf{F}_{si}\mathbf{H}_{aug}\mathbf{x}_{aug} \tag{5.3}$$

where $\mathbf{x}_{aug}=[\ \mathbf{x}^T\ \ \mathbf{u}^T\ ]^T$ and $\mathbf{H}_{aug}=[\ \mathbf{H}\ \ \mathbf{D}_z\ ]$ as presented in Section 3.4.4, $\mathbf{D}_{si}$ and $\mathbf{F}_{si}$ are the redistribution and failure matrices, respectively, for the i^{th} sensor failure. In this configuration, the MMAE-based controller redistributes both measurement and control signals during off-nominal flight, but in the nearly nominal region only control redistribution is used and the more accurate MMAE-generated state estimates are passed to the Block 40 FCS. Note that the ability to redistribute measurements is possible because of the acceleration measurements, which are linearly dependent on several other of the other measurements. Investigation into the performance of measurement redistribution is highly recommended for future research.

5.7 Real-Time, Man-in-the-Loop Simulation

While there are several aspects of control redistribution which remain to be investigated, a real-time simulation of even single failures would serve to validate the concept and to demonstrate its effectiveness in an even more realistic setting. A real-time simulation may be possible without resorting to a distributed processor system due to improvements made to the FORTRAN source code over the course of this research. The resulting eight second simulation uses less than six seconds of processor time on a Sun SPARC 20 computer. The six seconds of processor time accounts for the entire simulation, including aerodynamic updates and output operations to save the data files on disk, not just the controller. These numbers are promising for a true single-processor, real-time, man-in-the-loop implementation in the future.

5.8 In-flight Implementation

Although this chapter has highlighted many areas for future research, the suitability of MMAE-based redistribution, including both control and measurement redistribution, for a true in-flight implementation cannot be overlooked. Figure 5.1 shows the block diagram of the flight control system for such an implementation. Again, and this point is repeated for emphasis, the existing Block 40 FCS is retained in its entirety, ensuring that the basic structure remains the same. The additions are the MMAE for state and parameter estimation and the blocks for measurement and control redistribution. Not explicitly included in Figure 5.1 is the fact that, as mentioned in Section 5.6, during nearly nominal flight, the Block 40 FCS is provided the more accurate MMAE-generated estimates instead of the redistributed measurements.

Because both of the redistribution matrices reduce to the identity matrix (they simply pass through the measurements and control) for a fully functional aircraft, the control system has the exact same performance as the existing Block 40 FCS. Herein lies the true benefit of MMAE-based redistribution. First, during normal operation, the redistribution is completely transparent to the pilot, and the achieved performance is identical to that anticipated by the pilot from the Block 40 FCS. Second, in the event of a component failure, the redistribution ensures nearly the same performance is maintained for moderate inputs. So, at worst, the system gives the same performance as that already being flown on the VISTA F-16, and at best the system is able to compensate for failure conditions such that the pilot is able to continue the mission and return home.

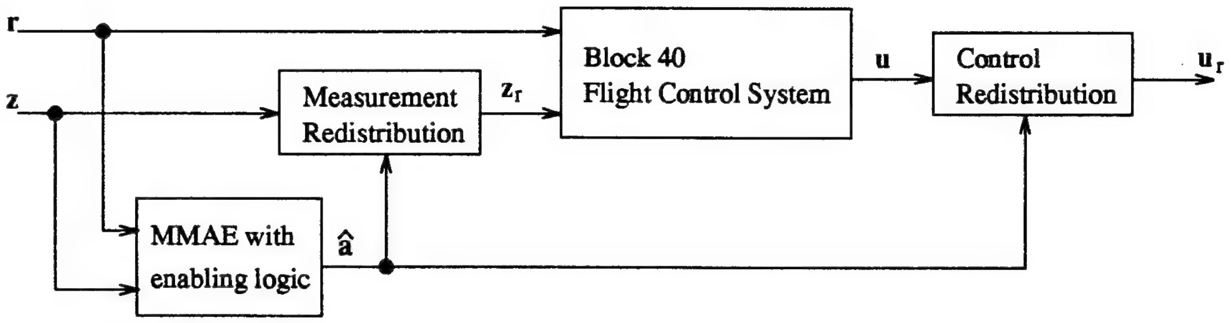


Figure 5.1 Possible In-flight Implementation of MMAE-based Redistribution

5.9 Chapter Summary

Although the MMAC using LQG/PI controllers did not yield an acceptable controller, the use of control redistribution resulted in an MMAE-based controller with excellent performance characteristics. All single failures were detected in less than one second, and the redistributed control nearly matched that of the fully functional aircraft. This chapter presented areas for future research and concluded with a description of a possible in-fight implementation of an MMAE-based redistribution controller.

Appendix A. MMAE-Based Control Redistribution Plots

This appendix contains comparisons made while verifying the performance of MMAE-based control redistribution on the SRF VISTA simulation. Description and analysis of the figures is found in Section 4.10.2.

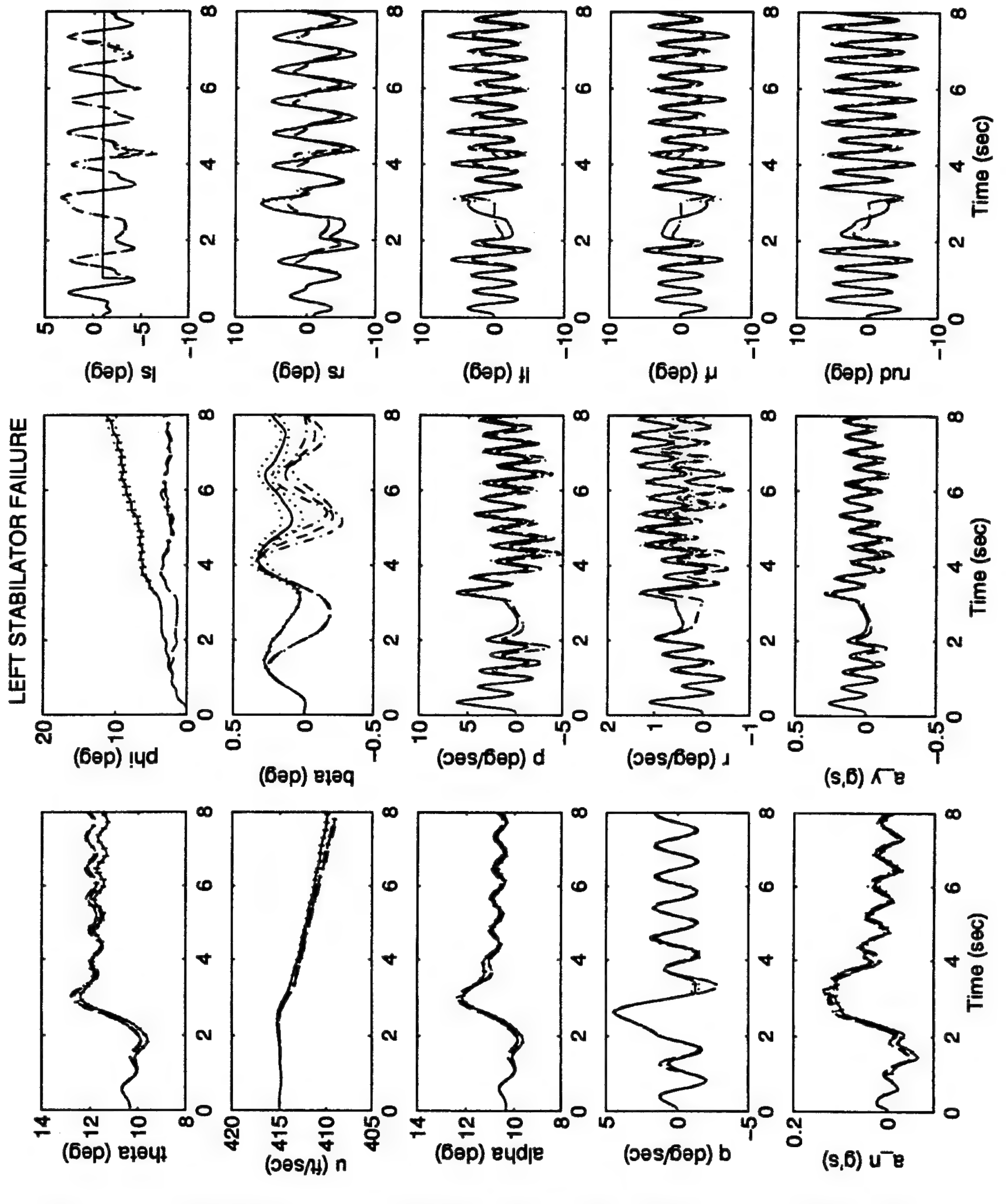


Figure A.1 MMAE-based Control Redistribution - Pitch Doublet - Left Stabilator Failure

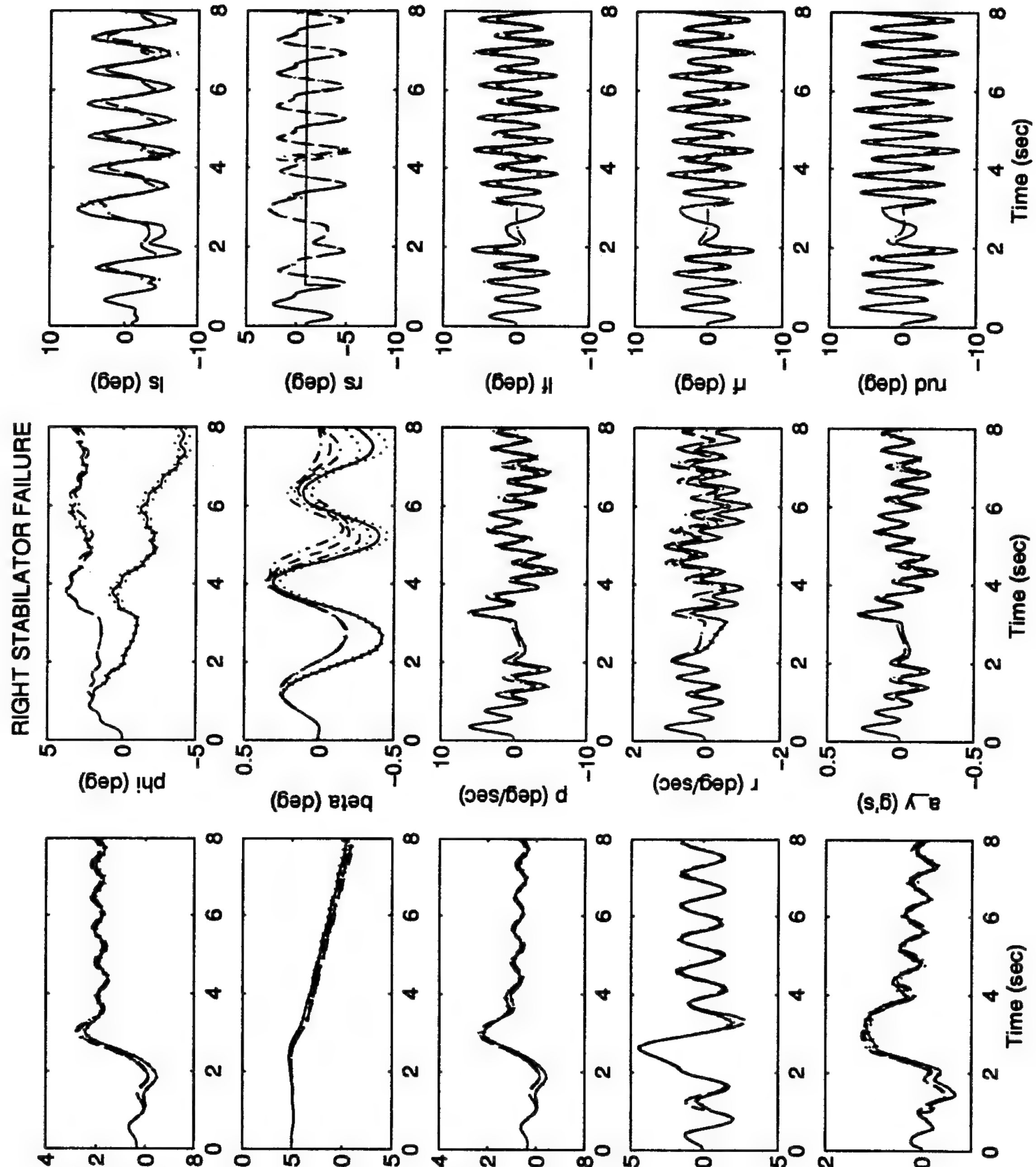


Figure A.2 MMAE-based Control Redistribution - Pitch Doublet - Right Stabilator Failure

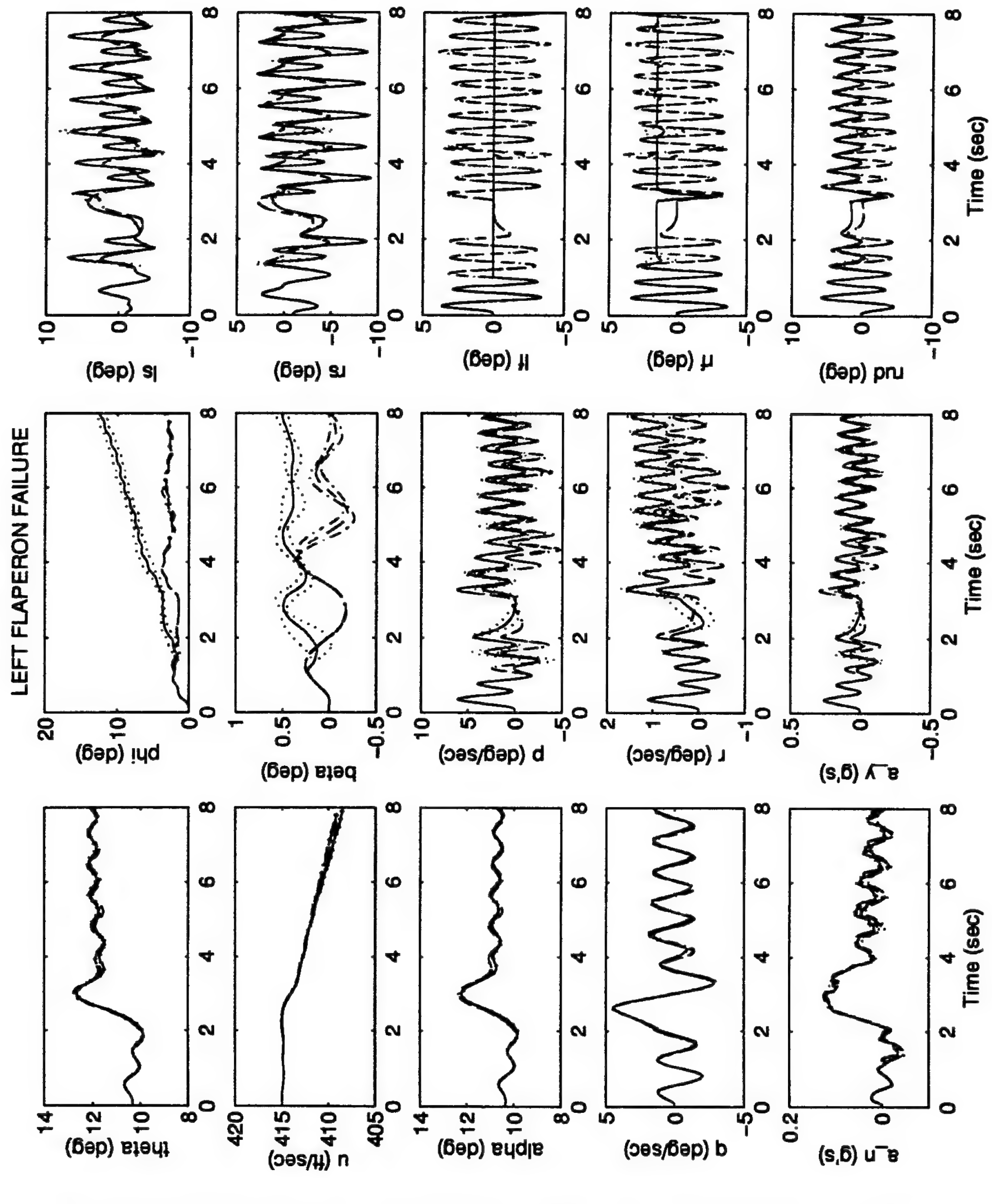


Figure A.3 MMAE-based Control Redistribution - Pitch Doublet - Left Flaperon Failure

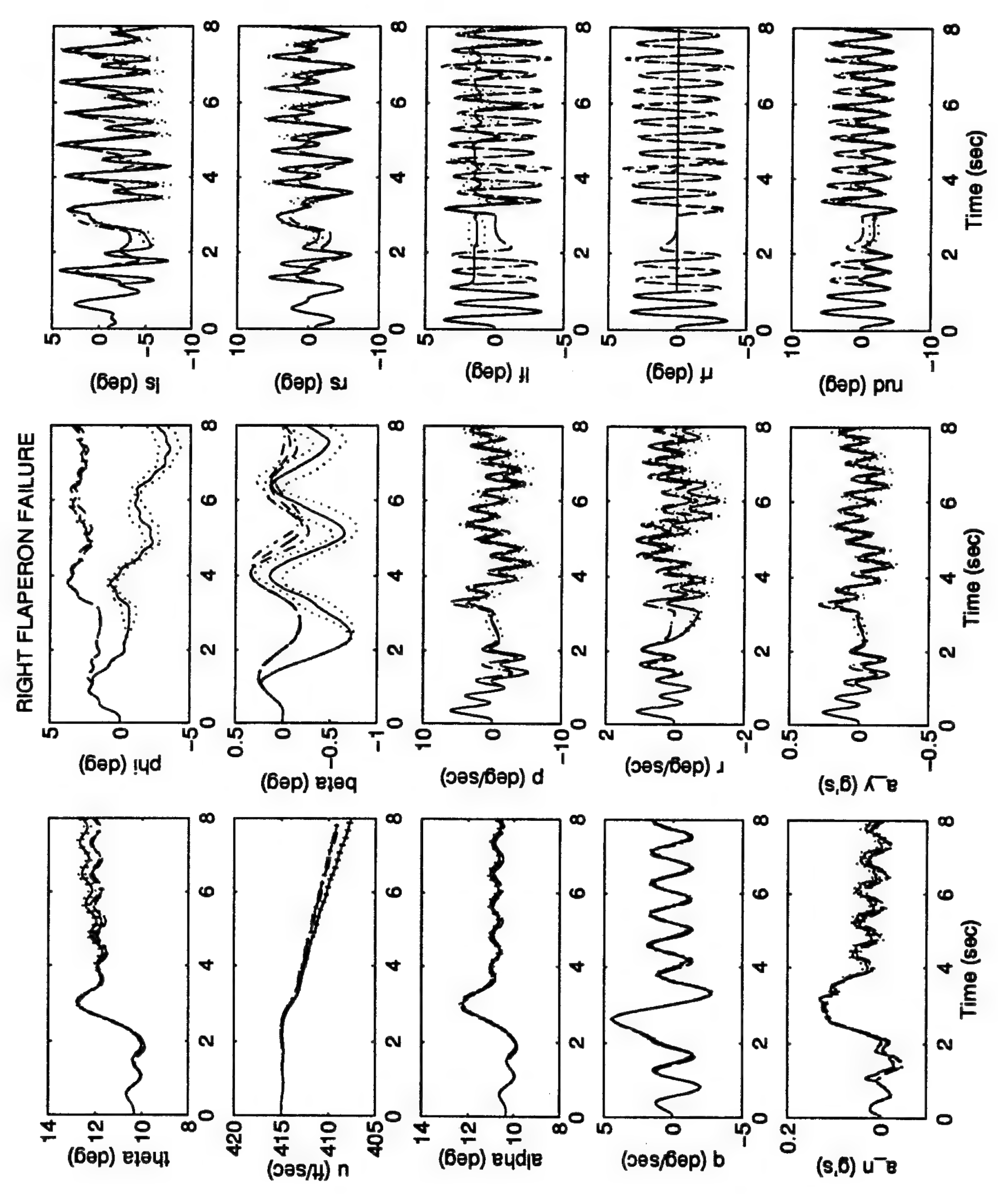


Figure A.4 MMAE-based Control Redistribution - Pitch Doublet - Right Flaperon Failure

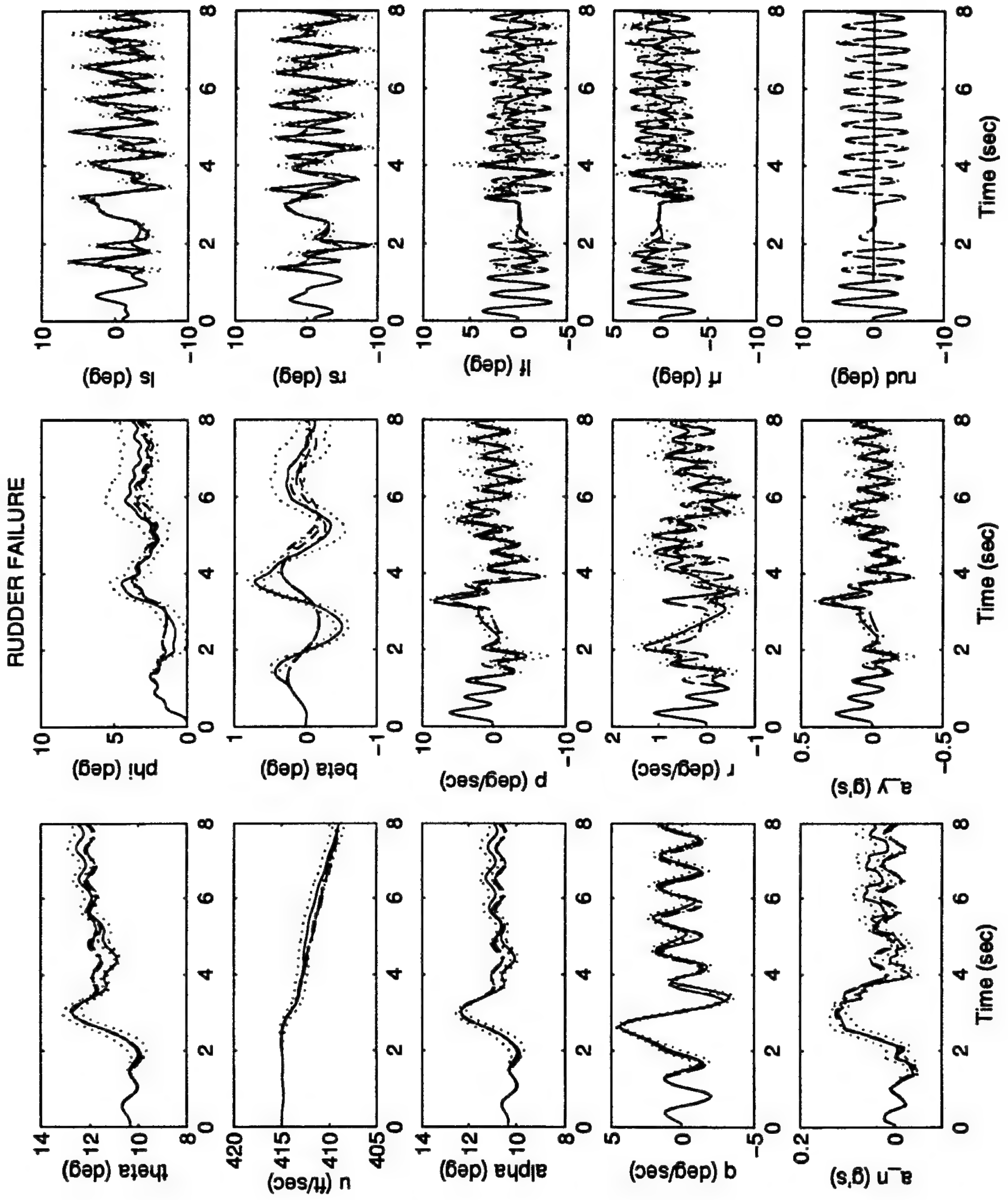


Figure A.5 MMAE-based Control Redistribution - Pitch Doublet - Rudder Failure

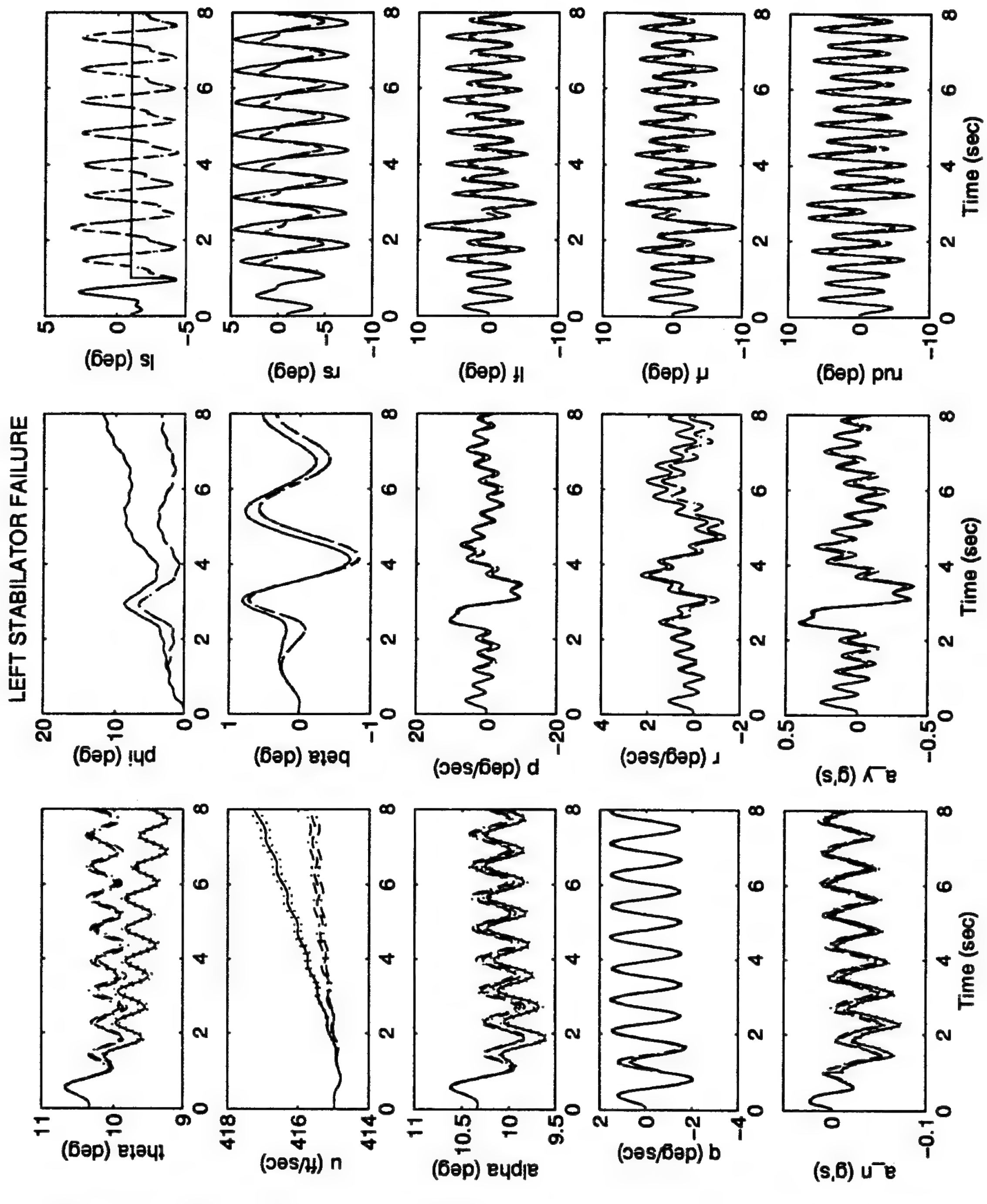


Figure A.6 MMAE-based Control Redistribution - Roll Doublet - Left Stabilator Failure

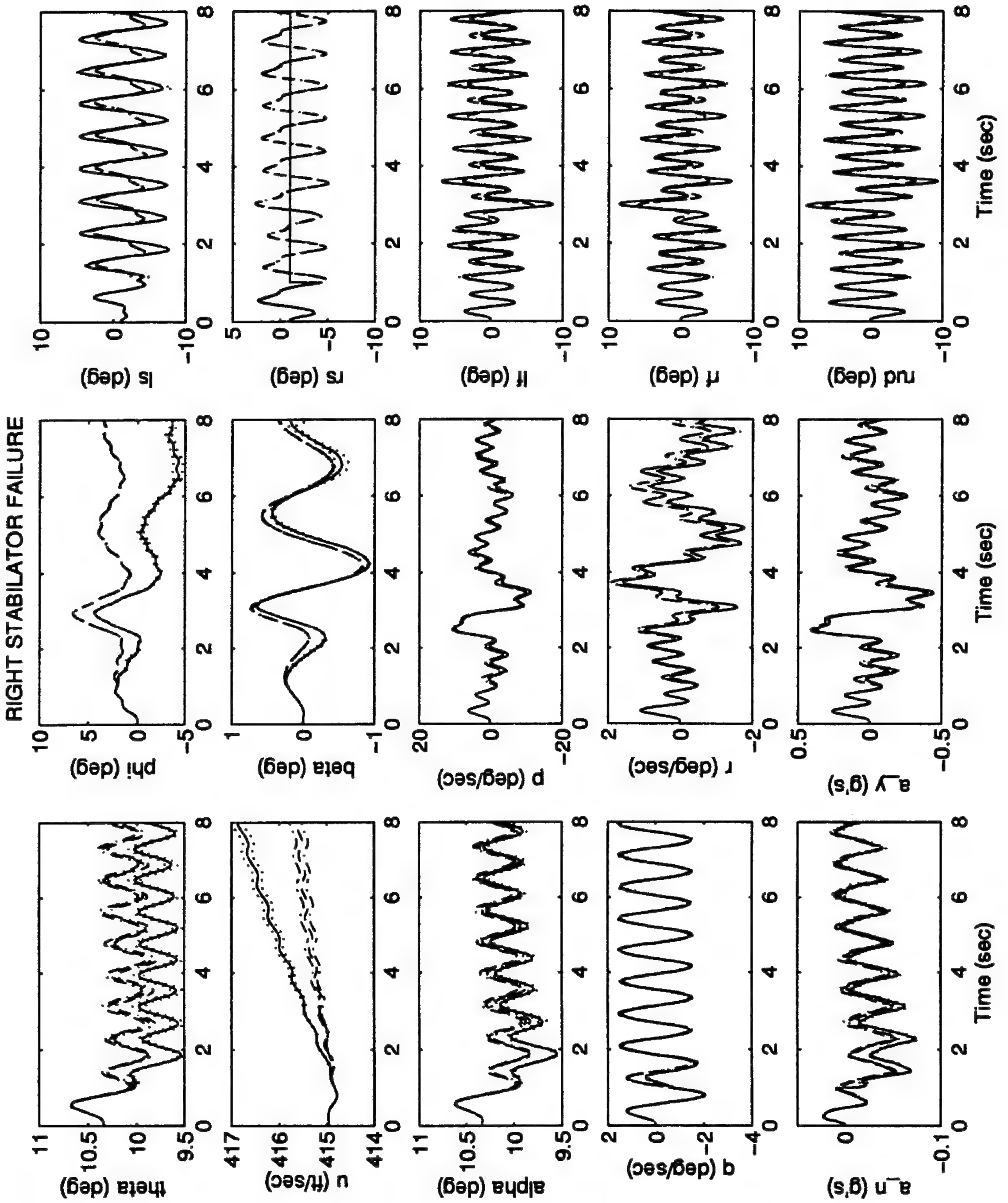


Figure A.7 MMAE-based Control Redistribution - Roll Doublet - Right Stabilator Failure

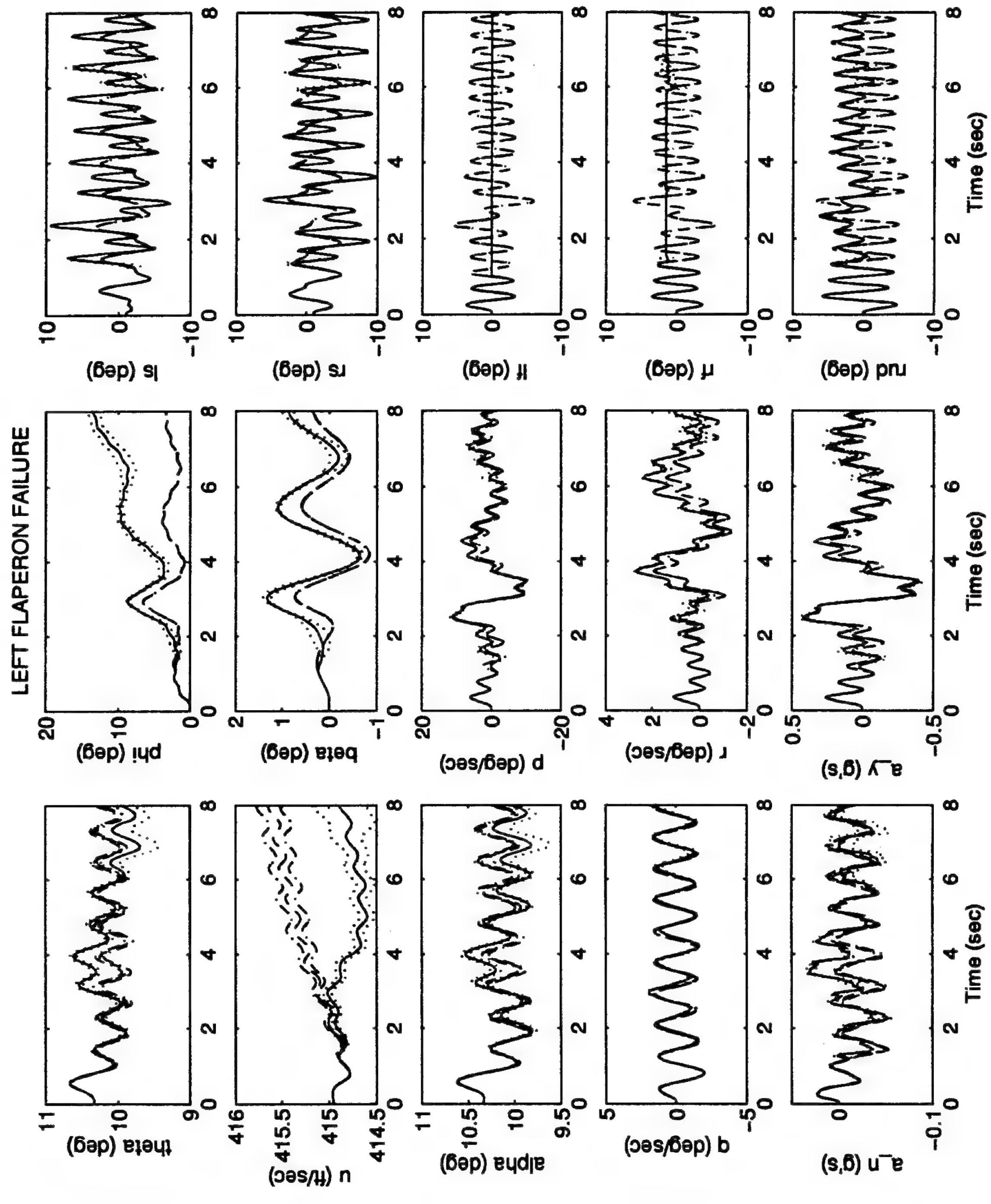


Figure A.8 MMAE-based Control Redistribution - Roll Doublet - Left Flaperon Failure

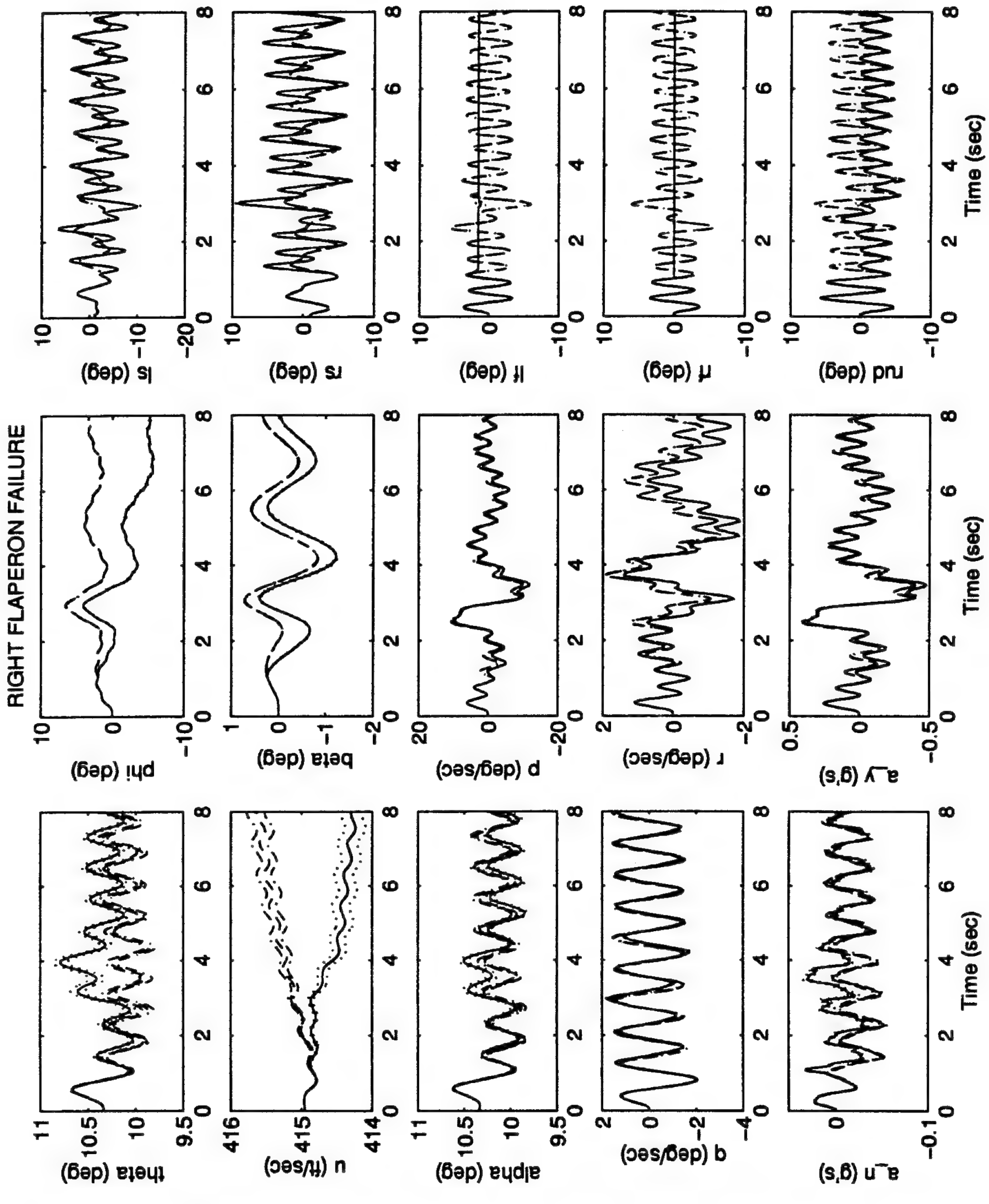


Figure A.9 MMAE-based Control Redistribution - Roll Doublet - Right Flaperon Failure

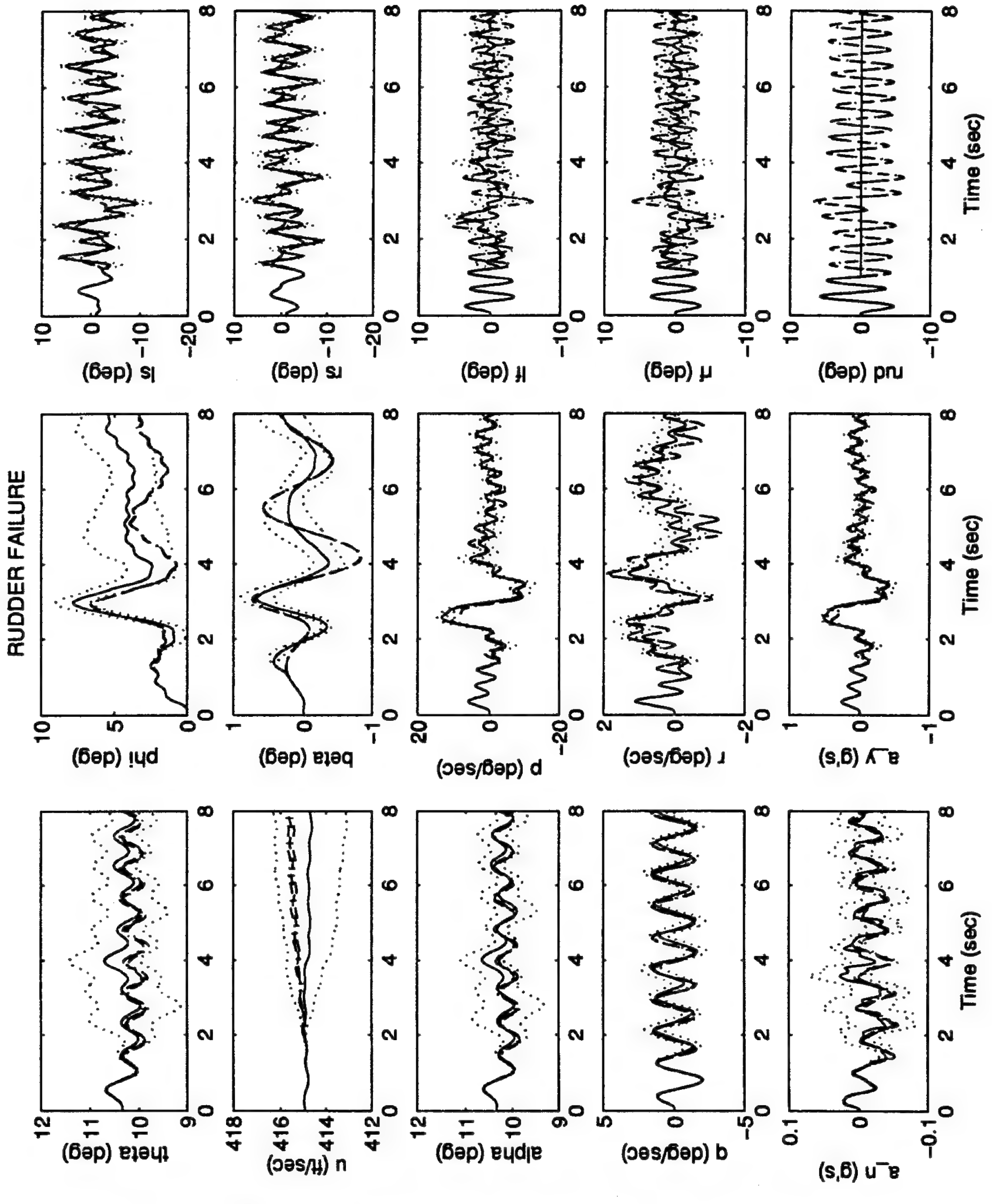


Figure A.10 MMAE-based Control Redistribution - Roll Doublet - Rudder Failure

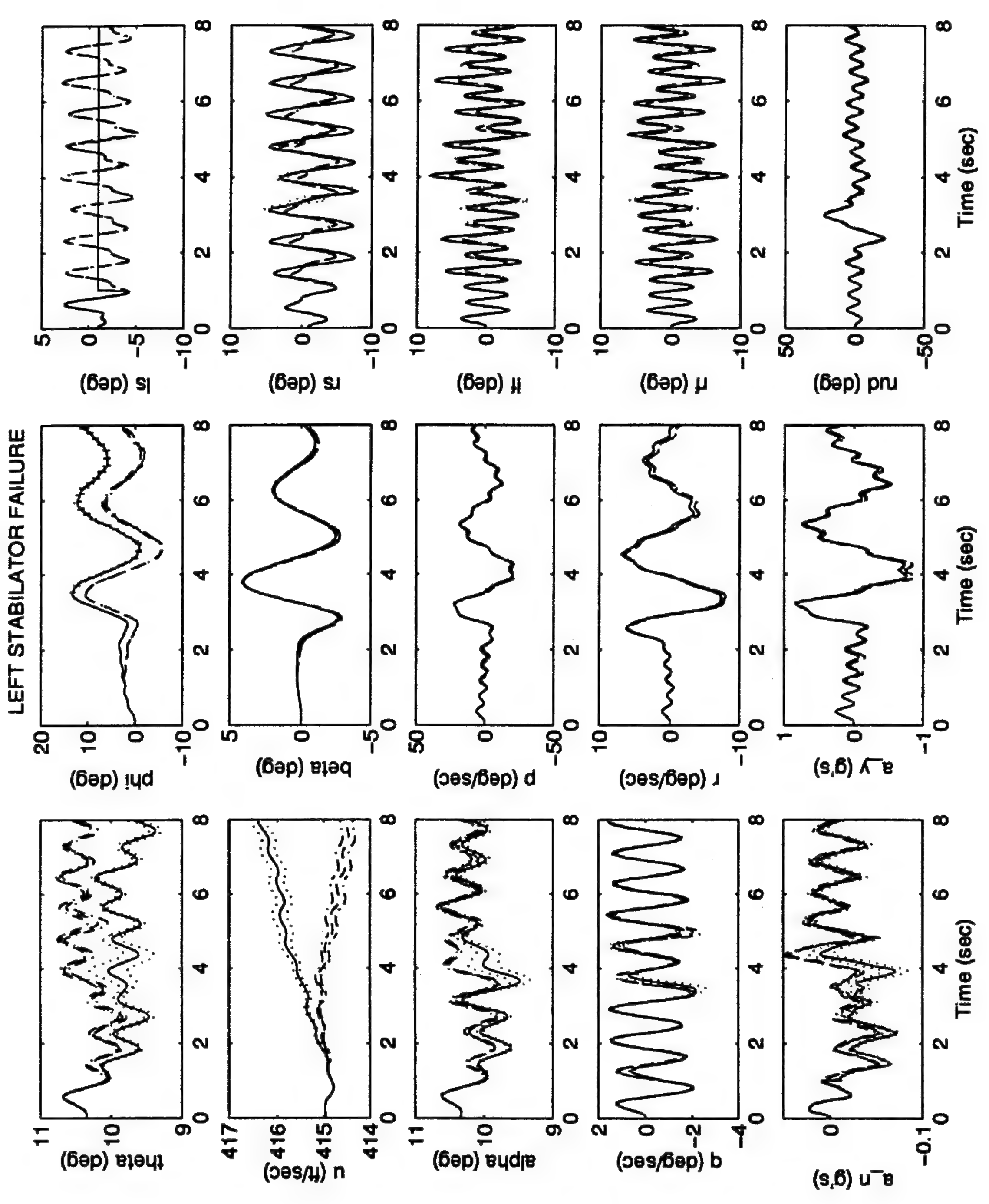


Figure A.11 MMAE-based Control Redistribution - Yaw Doublet - Left Stabilator Failure

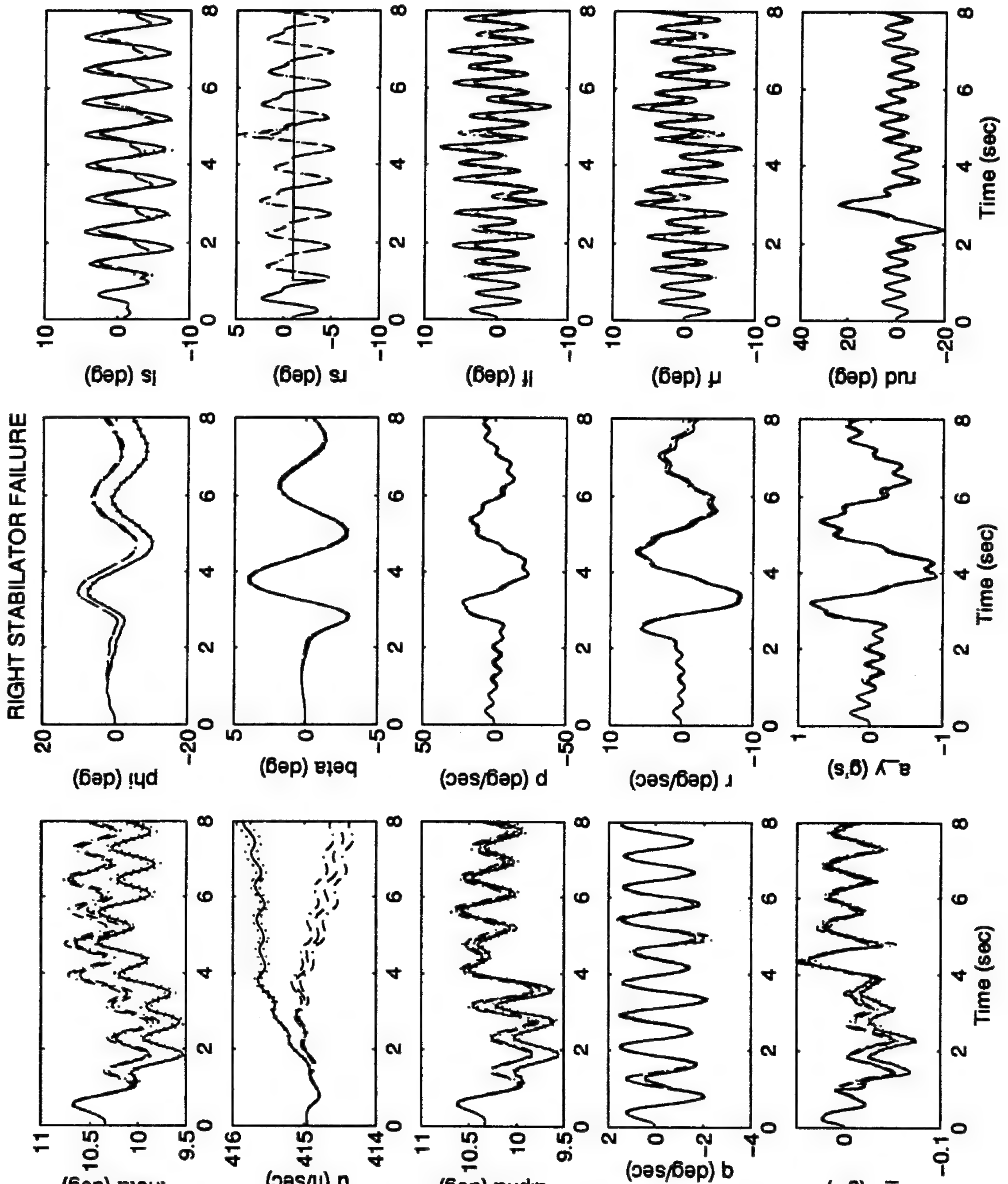


Figure A.12 MMAE-based Control Redistribution - Yaw Doublet - Right Stabilator Failure

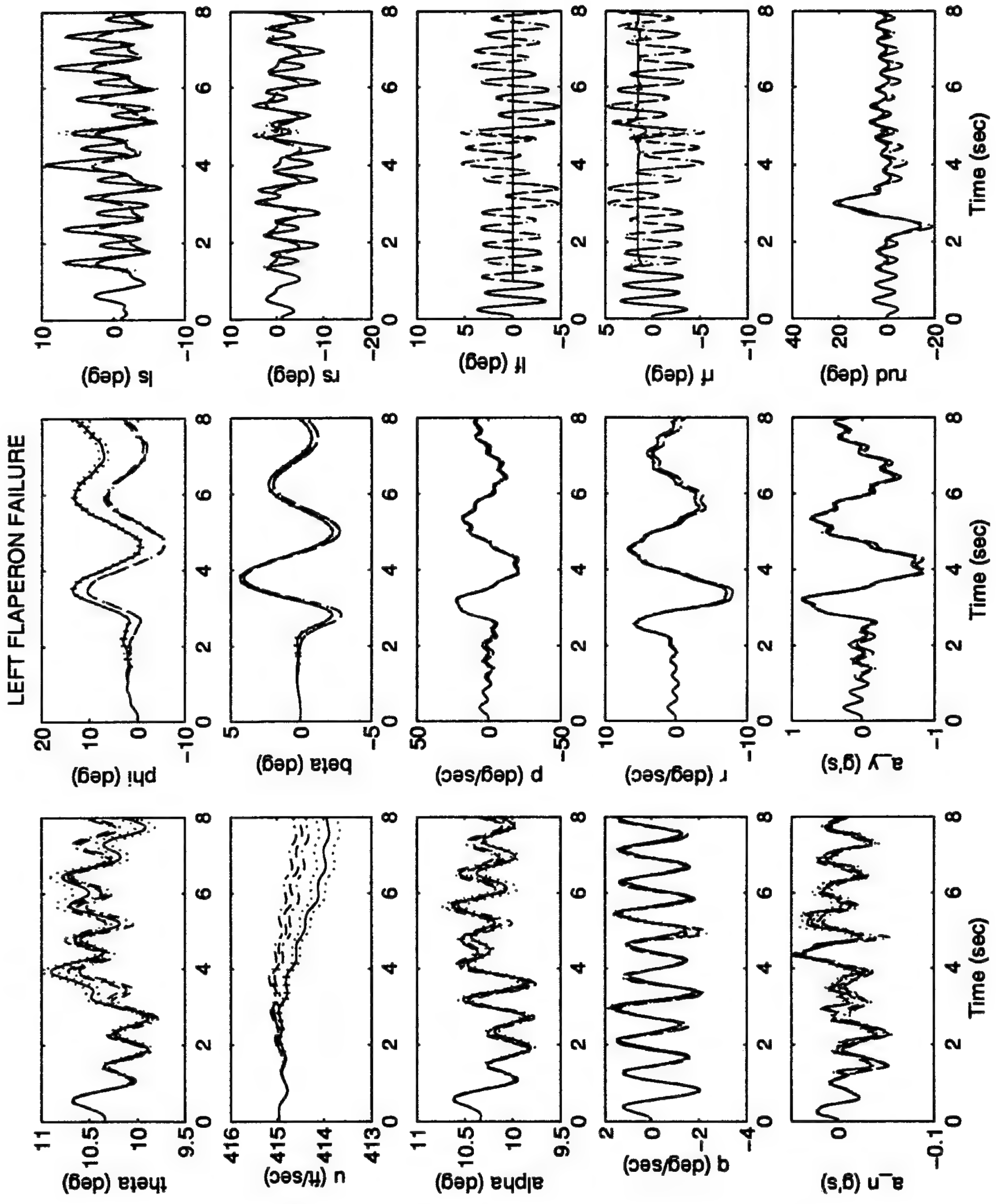


Figure A.13 MMAE-based Control Redistribution - Yaw Doublet - Left Flaperon Failure

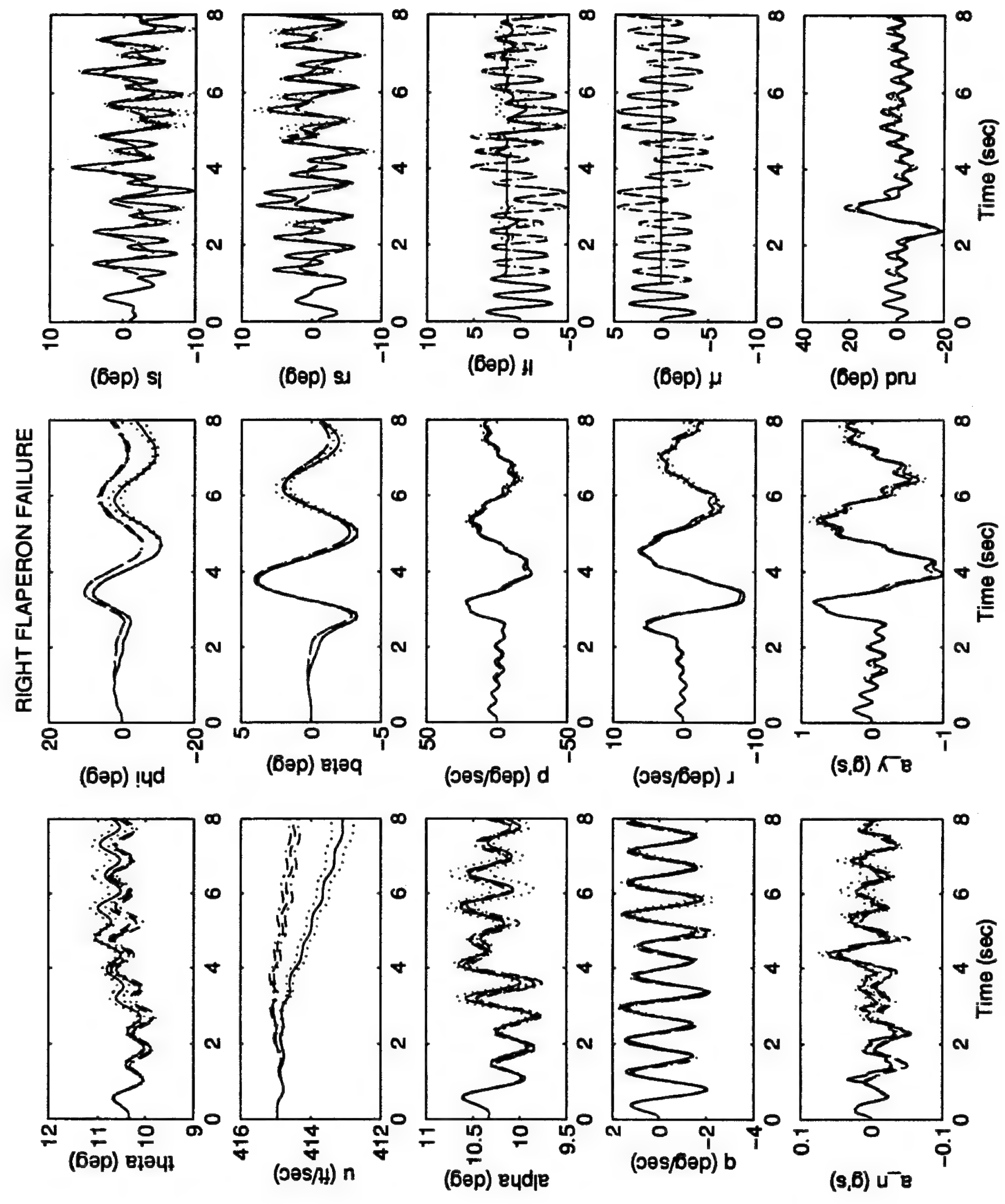


Figure A.14 MMAE-based Control Redistribution - Yaw Doublet - Right Flaperon Failure

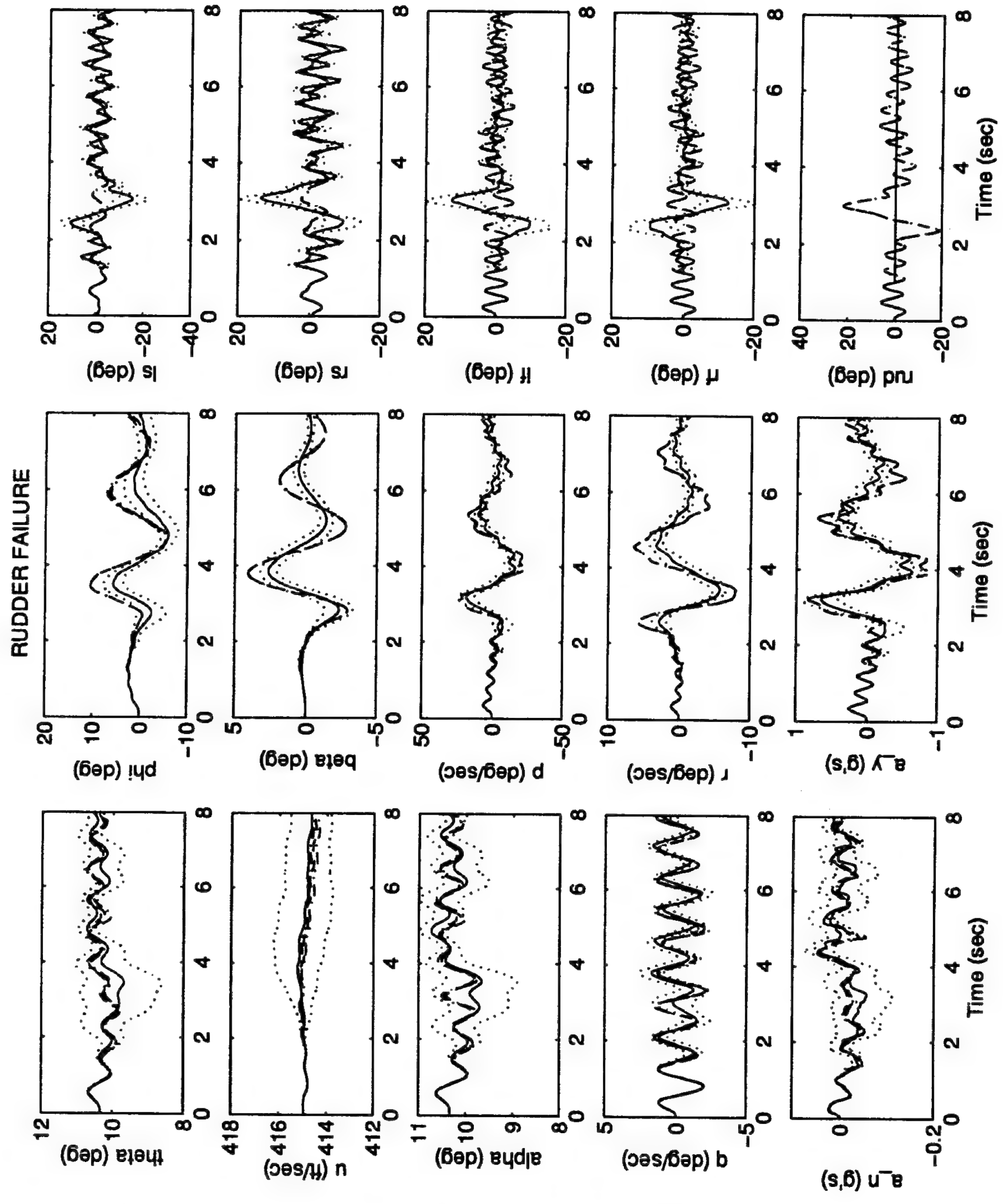


Figure A.15 MMAE-based Control Redistribution - Yaw Doublet - Rudder Failure

Appendix B. White Noise Approximation to the Dryden Wind Model

The white noise approximation to the Dryden wind model was first presented in the current form by Pogoda [42]. What follows is excerpted from a presentation of that derivation by Stratton [48] in Appendix A of his MS thesis. The text has been modified somewhat to reflect the differences in the research topics. Also Stratton makes many references to Martin's thesis [20] (which immediately preceded Stratton's research). Stratton's development, however, is complete in itself.

This appendix briefly repeats the results of approximating the Dryden wind model by zero-mean white Gaussian noise found in [20]. According to [20], the Dryden form of the spatial spectra for the turbulence velocities, as given in MIL-STD-1797A [36], must undergo two transformations to obtain the basic Dryden state model. First, the spatial forms of the Dryden model must be transformed to the temporal spectra forms, and second, the wind gust velocity terms (u, v, and w) must be converted into the angular state variables. the final resulting power spectral densities for the gust velocity, angle-of-attack, pitch rate, roll rate, sideslip angle, and yaw rate are expressed as [20]:

$$\Phi_{u_g}(\omega) = \sigma_u^2 \frac{2V_T}{L_u} \frac{1}{\omega^2 + (V_T/L_u)^2} \tag{B.1}$$

$$\Phi_{\alpha_g}(\omega) = \sigma_w^2 \frac{3}{2V_T L_u} \frac{\omega^2 + \frac{1}{12}(V_T/L_u)^2}{\left[\omega^2 + (V_T/L_u)^2\right]^2} \tag{B.2}$$

$$\Phi_{q_g}(\omega) = \left[\frac{\pi V_T}{4b}\right]^2 \frac{\omega^2}{\omega^2 + (\pi V_T/4b)^2} \Phi_{\alpha_g}(\omega) \tag{B.3}$$

$$\Phi_{p_g}(\omega) = \sigma_w^2 V_T \left[\frac{\pi^1 0}{128,000 b^7 L_w^2}\right]^{1/3} \frac{1}{\omega^2 + (\pi V_T/4b)^2} \tag{B.4}$$

$$\Phi_{\beta_g}(\omega) = \sigma_v^2 \frac{3}{2V_T L_v} \frac{\omega^2 + \frac{1}{12}(V_T/L_v)^2}{[\omega^2 + (V_T/L_v)^2]^2} \tag{B.5}$$

$$\Phi_{r_g}(\omega) = \left[\frac{\pi V_T}{3b}\right]^2 \frac{\omega^2}{\omega^2 + (\pi V_T/3b)^2} \Phi_{\beta_g}(\omega) \tag{B.6}$$

These power spectral density functions can be expressed equivalently as transfer functions which are driven by white noise, shown below:

$$G_{u_g}(s) = \frac{u_g(s)}{w_u(s)} = \sigma_u \left[\frac{2V_T}{L_u}\right]^{1/2} \frac{1}{s + (V_T/L_u)} \tag{B.7}$$

$$G_{\alpha_g}(s) = \frac{\alpha_g(s)}{w_w(s)} = \sigma_w \left[\frac{3}{2V_T L_u}\right]^{1/2} \frac{s + (V_T\sqrt{3}/6L_u)}{[s + (V_T/2L_u)]^2} \tag{B.8}$$

$$G_{q_g}(s) = \frac{q_g(s)}{w_w(s)} = \frac{\pi V_T}{4b} \frac{s}{s + (\pi V_T/4b)} G_{\alpha_g}(s) \tag{B.9}$$

$$G_{p_g}(s) = \frac{p_g(s)}{w_p(s)} = \sigma_w \left[\frac{\pi^1 0 V_T^3}{128,000 b^7 L_w^2}\right]^{1/6} \frac{1}{s + (\pi V_T/4b)} \tag{B.10}$$

$$G_{\beta_g}(s) = \frac{\beta_g(s)}{w_v(s)} = \sigma_v \left[\frac{3}{2V_T L_v}\right]^{1/2} \frac{s + (V_T\sqrt{3}/6L_v)}{[s + (V_T/L_v)]^2} \tag{B.11}$$

$$G_{r_g}(s) = \frac{r_g(s)}{w_v(s)} = \frac{-\pi V_T}{3b} \frac{s}{s + (\pi V_T/3b)} G_{\beta_g}(s) \tag{B.12}$$

where w_u, w_w, w_p and w_v are independent white noise sources of unit strength, b is the aircraft wing span, V_T is the aircraft velocity, L_u, L_v, and L_w are the turbulence scale lengths, and σ_u, σ_v, and σ_w are the turbulence intensities. For the medium/high-altitude model, $\sigma_u = \sigma_v = \sigma_w = \sigma$ and $L_u = 2L_v = 2L_w = 1750 feet$ [36]. The value of σ is based on light, moderate, or severe turbulence categories from Figure 262 of MIL-STD-1797A [36]. It should be noted that these turbulence specifications are numerical categories that represent greater turbulence than a pilot would normally expect from light, moderate, or severe designations [20].

The truth model of this thesis incorporates a zero-order wind model by approximating the time-correlated wind disturbances $u_g(t)$, $\alpha_g(t)$, $q_g(t)$, $p_g(t)$, $\beta_g(t)$, and $r_g(t)$ from the Dryden model with the white noises:

$$w(t) = \left[w_{u_g}(t) w_{\alpha_g}(t) w_{q_g}(t) w_{p_g}(t) w_{\beta_g}(t) w_{r_g}(t)\right]^T \tag{B.13}$$

where the white noise strengths are calculated by averaging the wind model power spectral density over the appropriate frequency range.

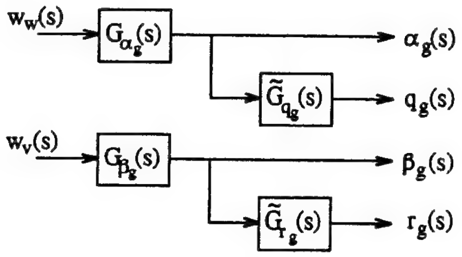


Figure B.1 Block Diagrams for Deriving Cross Spectral Densities

The white noise strengths become the diagonal elements of the **Q** matrix:

$$\mathbf{Q} = \begin{bmatrix} Q_{u_g} & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_{\alpha_g} & Q_{\alpha_g q_g} & 0 & 0 & 0 \\ 0 & Q_{\alpha_g q_g} & Q_{q_g} & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_{p_g} & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_{\beta_g} & Q_{\beta_g r_g} \\ 0 & 0 & 0 & 0 & Q_{\beta_g r_g} & Q_{r_g} \end{bmatrix} \tag{B.14}$$

The off diagonal elements of Equation B.14 are included since $\alpha_g(t)$ and $q_g(t)$ are both functions of $w_w(t)$, and similarly $\beta_g(t)$ and $r_g(t)$ are functions of $w_v(t)$. Therefore, to obtain the off-diagonal elements the cross spectral densities must be calculated. The cross spectral densities are defined by the relationships shown in Figure B.1, and by the equations:

$$\Phi_{\alpha_g q_g}(s) = \tilde{G}_{q_g}(s)\Phi_{\alpha_g}(s) \tag{B.15}$$

$$\Phi_{\beta_g r_g}(s) = \tilde{G}_{r_g}(s)\Phi_{\beta_g}(s) \tag{B.16}$$

where

$$\tilde{G}_{q_g}(s) = \frac{\pi V_T}{4b}\frac{s}{s+(\pi V_T/4b)} \tag{B.17}$$

$$\tilde{G}_{r_g}(s) = \frac{-\pi V_T}{3b}\frac{s}{s+(\pi V_T/3b)} \tag{B.18}$$

Substituting s/j for w in Equations (B.2) and (B.5), and substituting the the results into Equations (B.15) and (B.16), respectively, yields the cross spectral density equations in Laplace s notation:

$$\Phi_{\alpha_g q_g}(s) = \frac{\pi\sigma_w^2}{32bL_w}\frac{s}{s+(\pi V_T/4b)}\frac{(V_T/L_w)^2 - s^2}{[(V_T/2L_w)^2 - s^2 +]^2} \tag{B.19}$$

$$\Phi_{\beta_g r_g}(s) = \frac{-\pi\sigma_v^2}{24bL_v}\frac{s}{s+(\pi V_T/3b)}\frac{(V_T/L_v)^2 - s^2}{[(V_T/2L_v)^2 - s^2 +]^2} \tag{B.20}$$

Normally, the white noise strength approximation of a power spectral density function can be found by obtaining the average magnitude over an appropriate frequency range from its Bode plot. However, the cross spectral densities in Equations (B.19) and (B.20) include a complex component, i. e. the white noise strengths cannot be simply obtained by the above method. Another method to obtain the noise strengths is to take the inverse Laplace (or Fourier, as the case may be) transform of Equations (B.19) and (B.20). From the transform, obtain the magnitude of the cross-correlation at $tau = 0$, and denote this value as σ^2; then the height of the power spectral density curve (white noise strength approximation) is $2\sigma^2 T$, where T is the time constant of the first order transfer function [20].

Another and perhaps easier method to obtain the cross spectral densities is first to approximate the power spectral densities of Equations (B.2) and (B.5) as constant heights, which corresponds to the white noises $w_{\alpha g}(t)$ and $w_{\beta g}(t)$, respectively. The magnitudes of these power spectral densities can then be used as inputs to the transfer functions described in Equations (B.17) and (B.18) to approximate the cross spectral densities [20].

Repeated from [20], shown in Table B.1, is the bandwidth for each of the aircraft state variables and the noise strengths based on a wind turbulence RMS value of σ=1 ft/sec. Although the value of σ=1 falls into the category of light turbulence in MIL-STD-1797A, it more accurately represents light to moderate turbulence [20]. The white noise strengths can easily be scaled for increased (or decreased) RMS wind turbulence values, since each power spectral density equation contains a σ^2 term. Note that the values in Table B.1 are based on an aircraft velocity of 622 ft/sec (0.6 Mach, 20,000 ft); the aircraft velocity used in this thesis is 415 ft/sec (0.4 Mach, 20,000 ft). Although the difference in velocity results in some difference in the average noise strength, because Eide [8] chose to use these same values when he incorporated the Dryden wind model into the SRF VISTA simulation, the same values are carried over into the current MMAC research. (The only exception is that the $\mathbf{Q}(1,1)$ term is scaled as described in Section 3.4.3 to account for the fact that the resulting noise is not typically described as light turbulence).

Table B.1 Dryden Wind Model Noise Strength Approximations

Variable	Aircraft Bandwidth	Units	Average Noise Strength	Units
u	0.25	$\frac{rad}{sec}$	4.5	$\frac{ft^2}{rad \cdot sec}$
α	4.0	$\frac{rad}{sec}$	3.0^{-6}	$rad \cdot sec$
q	20.0	$\frac{rad}{sec}$	1.5×10^{-6}	$\frac{rad}{sec}$
α vs q	4.0	$\frac{rad}{sec}$	1.1×10^{-8}	rad^2
p	15.0	$\frac{rad}{sec}$	6.0×10^{-6}	$\frac{rad}{sec}$
β	3.5	$\frac{rad}{sec}$	3.0×10^{-6}	$rad \cdot sec$
r	7.0	$\frac{rad}{sec}$	2.4×10^{-6}	$\frac{rad}{sec}$
β vs r	3.5	$\frac{rad}{sec}$	6.3×10^{-9}	rad^2

The white noise strength approximations given in Table B.1 are then put into the appropriate locations of the Q matrix as shown in Equation (B.14). the white noise approximations are incorporated into the 8th-order aircraft model through the matrix G, as derived in [20] and defined in Equation (B.21) in terms of the primed aircraft dimensional derivatives, as defined by [20] and used in this thesis.

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ X'_u & X'_\alpha & X'_q & 0 & 0 & 0 \\ Z'_u & Z'_\alpha & Z'_q & 0 & 0 & 0 \\ M'_u & M'_\alpha & M'_q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Y'_p & Y'_\beta & Y'_r \\ 0 & 0 & 0 & L'_p & L'_\beta & L'_r \\ 0 & 0 & 0 & N'_p & N'_\beta & N'_r \end{bmatrix} \tag{B.21}$$

Appendix C. SRF Modification Table

The SRF VISTA simulation [15, 16] is provided by Wright Laboratories Flight Dynamics Directorate. The source code is written in FORTRAN and setup for compilation on Sun SPARC-stations. The MMAC code is based on the *MMAESIM* software written by Menke [34]. Eide [8] incorporated this *MMAESIM* code into the SRF VISTA simulation. Eide also added improved sensor/actuator failure models, a zero-order Dryden wind model[1], and the linear lateral acceleration model to the SRF VISTA. This software was further modified to accommodate the additional controllers required for an MMAC. Tables C.1 and C.2 summarize the modified SRF files and the additional MMAC files, respectively, needed to convert the original SRF VISTA into the version used in this research.

Table C.1 Modified SRF Files

Filename	Description of modification
afm.F	- Replace A_y calculation with linear model - Add full Dryden wind model (unused)
atr40.F	- Add actuator failure model
buildmx.F	- Modify to generate MATLAB readable output file - Add $\mathbf{G}$, $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{B}_{mod}$
control.F	- Replace *FCS_IO* call with *MMAC* function call
fcs_io.F	- Replace full-state feedback with state estimates - Add control redistribution - Add LQG function call
getcom.F	- Add dither
quat.F	- Add zero-order Dryden wind model effects
sensors.F	- Add sensor failure model - Add sensor noise corruption
turb2.F	- Add Dryden rotational gusts P,Q,R (unused)
vista.F	- Initialize MMAC - Add file I/O headers and trailers
wind.F	- Replace SRF wind model with zero-order Dryden model
wind_ic.F	- Remove call to SRF wind model
wind_setup.F	- Add zero-order Dryden wind model states - Add Dryden wind model rotational states (unused)
datapool.inc	- Add (unused) variables for full Dryden wind model
wind.inc	- Add zero-order Dryden wind model states - Add Dryden wind model rotational states (unused)

[1] A full Dryden wind model was also implemented, though it has not yet been tested and is not used in this research.

Table C.2 Additional MMAC Files

Filename	Description of contents
failchk.F	- Evaluate probabilities and declare failures as appropriate
fltreq.F	- Update and propagate the Kalman filters
getflags.F	- Read in flags concerning operating and failure status
inimmae.F	- Read in filter/controller data
lqg.F	- Implement the LQG controllers
matfunc.F	- Library of matrix algebra functions
mmac.F	- Top level function for the MMAC algorithm
mmreset.F	- Reset filter bank after a bank swap
nommmae.F	- Store nominal values of pertinent variables for use by the MMAC
probeval.F	- Blend the estimates and control laws from the elemental filter/controllers
sort.F	- Quicksort used by probeval to implement lower bounds
mmac.inc	- Declare variables and parameters for use by the MMAC

For convenience, the modified SRF VISTA version allows one of four FCS modes to be selected via software flags read into the simulation at run time. These modes are:

1. Block 40 FCS
2. MMAE-based controller using the Block 40 FCS
3. MMAE-based control redistribution
4. MMAC using the Block 40 FCS (if desired) and LQG controllers

All four FCS modes incorporate dither, sensor noise, and the linear lateral acceleration model. The software flags, contained in file *FLAGS.DAT*, also select the failure modes and times of failures, and enable/disable the zero-order Dryden wind model and bank swapping. The location of the filter and controller files is also provided to the SRF VISTA via *FLAGS.DAT*.

Other simulation parameters, such as flight condition and piloted commands, are entered via a name.*par* file created by the Transportable Applications Executive (TAE) [16]. To facilitate the reproduction of the results contained in this thesis, the parameters used in this research are given in Table C.3, reproduced from [8]. The first column of this table lists the variable as presented by the TAE, the second column provides a brief description of that variable (the online help provides more detailed descriptions), and the third column gives the values exactly as entered into the TAE.

Table C.3 SRF Parameter Values

Parameter	Description	Value
outfile	Name of APRET output file	"vista.output"
ofiletyp	Output file format	"apret"
eqmot	Equation of motion to be used	"gd"
stoptime	Simulation duration (sec)	8.0
saverate	Data recording sample rate (Hz)	32.0
afmrate	Airframe model iteration rate	128.0
DFCSrate	DFCS iteration rate	64.0
trimic	Trim with initial conditions	"no"
ctlfile	Name for SEL CTL file	(null value)
altitude	Initial altitude (ft)	20000
Mach	Initial airspeed	0.4
Xpos	Initial aircraft X axis (ft)	0.0
Ypos	Initial aircraft Y axis (ft)	0.0
heading	Initial A/C true heading (deg)	0.0
gamma	Initial glide slope angle (deg)	0.0
thrcntrl	Throttle control type	"constant"
stores	Stores configuration	"standard"
fltcond	Flight Condition	"up+away"
atmos	Type of day for atmosphere model	"standard"
cmdfile	Name of command file	(null value)
commands	Pilot commands	(null value)
cmdmagn	Command magnitude	(null value)
cmdtime	Start time for each command	(null value)
cmddur	Duration of each command	(null value)
gear	Initial landing gear setting	(null value)
geartime	Time to toggle gear setting	(null value)
spdbrake	Initial speed brake setting	(null value)
spdtime	Time to toggle speed brake	(null value)
flap	Initial flap setting	(null value)
flaptime	Time to toggle flap setting	(null value)
mxfile	Name of MATRIXx file for linear model generation	"srfdat.m"
multwgt	A/C weight multiplication factor for linear model generation	1.0
lse	Local slope extraction	"no"
saveit	Debug data save flag	"no"
savefn	Debug data save filename	"debug_iofn.dat"
restorit	Debug data restore flag	"no"
restorfn	Debug data restore filename	"debug_iofn.dat"

Bibliography

1. A., John and Peter S. Maybeck. "Flexible Spacestructure Control Via Moving-Bank Multiple Model Algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, *30*:750–757 (July 1994).
2. Athans, M., *et al.* "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method – Part I: Equilibrium Flight," *IEEE Transactions on Automatic Control*, *AC–22*(5):768–780 (October 1977).
3. Blakelock, John H. *Automatic Control of Aircraft and Missiles.* New York: John Wiley & Sons, Inc., 1991.
4. Broussard, J. R. and P. W. Berry. "The Relationship between Implicit Model Following and Eigenvalue-Eigenvector Placement," *IEEE Transactions on Automatic Control*, *AC–25*(5):591–594 (1980).
5. Cacciatore, 2Lt Vincent J. *A Quantitative Feedback Theory FCS Design for the Subsonic Envelope of the VISTA F-16 Including Configuration Variation and Aerodynamic Control Effector Failures.* MS Thesis, AFIT/GE/ENG/95D-04, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1995.
6. Chang, C. B. and M. Athans. "State Estimation for Discrete Systems with Switching Parameters," *IEEE Transactions on Aerospace and Electronic Systems*, *AES–14*(4):418–424 (May 1978).
7. Dasgupta, S. and L. C. Westphal. "Convergence of Partitioned Adaptive Filters for Systems with Unknown Biases," *IEEE Transactions on Automatic Control*, *28*:614–615 (May 1983).
8. Eide, Capt Peter Keith. *Implementation and Demonstration of a Multiple Model Adaptive Estimation Failure Detection System for the F-16.* MS Thesis, AFIT/GE/ENG/94D-06, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1994.
9. Elliott, Jarrell R. "NASA's Advanced Control Law Program for the F-8 Digital Fly-by-Wire Aircraft," *IEEE Transactions on Automatic Control*, *22(5)*:753–757 (October 1977).
10. Fry, C. M. and A. P. "On Hierarchical Structure Adaptation and Systems Identification," *International Journal of Control*, *20(3)*:433–452 (September 1974).
11. General Dynamics, Fort Worth Division, "Dwg 711ZC001 - Rev A," September 1990.
12. Gierke, Henning E., et al. *Stochastic Models, Estimation, and Control, II, Book 1*, 355–405. Washington, D.C.: Scientific and Technical Information Office, National Aeronautics and Space Administration, 1975.
13. Gilbert, Elmer G. "Conditions for Minimizing the Norm Sensitivity of Characteristic Roots." *Proceedings of the IEEE Conference on Decision and Control22*. 325–330. 1983.
14. Hawkes, R. M. and J. B. Moore. "Performance Analysis of Bayesian Parameter Estimators," *Proceedings of the IEEE*, *64*:1143–1150 (August 1976).
15. Inc., Century Computing. *Simulation/Rapid-Prototyping Facility (SRF) F-16 VISTA Simulation Engineer's Guide*, December 1992.
16. Inc., Century Computing. *Simulation/Rapid-Prototyping Facility (SRF) F-16 VISTA Simulation User's Manual*, October 1992. Draft Sun/UNIX version.
17. Lainiotis, D. G. "Partitioning: A Unifying Framework for Adaptive Systems, I: Estimation," *Proceedings of the IEEE*, *64*:1182–1197 (August 1976).
18. Lane, Captain David W. *Mutiple Model Adaptive Estimation Applied to the LAMBDA URV for Failure Detection and Identification.* MS Thesis, AFIT/GE/ENG/93D-23, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.

19. Magill, D. T. "Optimal Adaptive Estimation of Sample Stochastic Processes," *IEEE Conference on Decision and Control, AC-10*(4):434–439 (October 1965).

20. Martin, Capt Richard Maurice. *LQG Synthesis of Elemental Controllers for AFTI/F-16 Adaptive Flight Control*. MS Thesis, AFIT/GE/ENG/90D-36, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1990.

21. The MathWorks, Inc. *MATLAB (registered trademark)* (Version 4.2c Edition), November 1994.

22. Maybeck, P. S. and R. I. Suizu. "Adaptive Tracker Field of View Variation Via Multiple Model Filtering," *IEEE Transactions on Aerospace and Electronic Systems, 21(4)*:529–537 (July 1985).

23. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, I*. New York: Academic Press, Inc., 1979.

24. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, III*. New York: Academic Press, Inc., 1982.

25. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, II*. New York: Academic Press, Inc., 1982.

26. Maybeck, Peter S. and Capt Peter D. Hanlon. "Performance Enhancement of a Multiple Model Adaptive Estimator," *IEEE Conference on Decision and Control* (1993).

27. Maybeck, Peter S. and Karl P. Hentz. "Investigation of Moving-Bank Multiple Model Adaptive Algorithms," *Journal of Guidance and Control, 10(1)*:90–96 (January–February 1987).

28. Maybeck, Peter S., et al. "Target Tracking Using Infrared Measurements and Laser Illumination," *IEEE Transactions on Aerospace and Electronic Systems, 30*:758–768 (July 1994).

29. Maybeck, Peter S. and Donald L. Pogoda. "Multiple Model Adaptive Controller for the STOL F-15 with Sensor/Actuator Failures," *Proceedings of the 28th Conference on Decision and Control*, 1566–1572 (December 1989).

30. Maybeck, Peter S. and Michael R Schore. "Reduced-Order Multiple Model Adaptive Controller for Flexible Spacestructure," *IEEE Transactions on Aerospace and Electronic Systems, 28*:756–767 (July 1992).

31. Maybeck, Peter S. and Richard D. Stevens. "Reconfigurable Flight Control Via Multiple Model Adaptive Control Methods," *IEEE Transactions on Aerospace and Electronic Systems*, 470–480 (May 1991).

32. Maybeck, Peter S. William G. Miller and Jean M. Howey. "Robustness Enhancement for LQG Digital Flight Controller Design," *Proceedings of the IEEE National Aerospace and Electronic Conference, Dayton, OH*, 518–525 (May 1984).

33. Maybeck, Peter S., Professor, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH. Personal Interviews. January–October, 1995.

34. Menke, Timothy E. *Multiple Model Adaptive Estimation Applied to the VISTA F-16 with Actuator and Sensor Failures*. MS Thesis, AFIT/GA/ENG/92J-01, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, June 1992.

35. Menke, Timothy E. and Peter S. Maybeck. "Sensor/Actuator Failure Detection in the VISTA F-16 by Multiple Model Adaptive Estimation," *Proceedings of American Control Conference, San Francisco, June 1993* (1993).

36. *MIL-STD-1797A Flying Qualities of Piloted Aircraft*, January 1990.

37. Moore, J. B. and R. M. Hawkes. "Decision Methods in Dynamic System Identification." *Proceedings of the IEEE Conference on Decision and Control14*. 645–650. 1975.

38. Nelson, Robert C. *Flight Stability and Automatic Control.* New York: McGraw-Hill, Inc., 1989.

39. Nise, Norman S. *Control Systems Engineering.* New York: The Benjamin/Cummings Publishing Company, Inc., 1992.

40. Payson, Capt Steven S. *Flight Control System for the CRCA Using a Command Generator Tracker with PI Feedback and Kalman Filter.* MS Thesis, AFIT/GE/ENG/89M-6, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, March 1989.

41. Phillips, Maj Scott N. *A Quantitative Feedback Theory FCS Design for the Subsonic Envelope of the VISTA F-16 Including Configuration Variation.* MS Thesis, AFIT/GE/ENG/94D-24, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1994.

42. Pogoda, Capt Donald L. *Multiple Model Adaptive Controller for the STOL F-15 with Sensor/Actuator Failures.* MS Thesis, AFIT/GE/ENG/88D-23, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1988.

43. Reid, Gary J. *Linear Systems Fundamentals.* New York: McGraw-Hill, Inc., 1983.

44. Ridgely, Brett D., Professor, Department of Aeronautics and Astronautics Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH. Personal Interview. September, 1995.

45. Schaefer, Karl E. *Bioastronautics.* New York: The Macmillan Company, 1964.

46. Stevens, Capt Richard D. *Characterization of a Reconfigurable Multiple Model Adaptive Controller Using A STOL F-15 Model.* MS Thesis, AFIT/GE/ENG/89D-52, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1989.

47. Strang, Gilbert. *Linear Algebra and Its Applications* (third Edition). San Diego: Harcourt Brace Jovanovich, Publishers, 1988.

48. Stratton, Capt Gregory L. *Actuator and Sensor Failure Identification using a Multiple Model Adaptive Technique for the VISTA/F-16.* MS Thesis, AFIT/GE/ENG/91D-53, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1991.

49. Tugnait, J. K. "Comments on "State Estimation for Discrete Systems with Switching Parameters"," *IEEE Transactions on Aerospace and Electronic Systems*, *AES-15*(3):464 (May 1979).

50. WL/FIGXF, WPAFB, OH, "VISTA/NF-16 Technical Data for Customer Usage," August 1991.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 1995	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE MULTIPLE MODEL ADAPTIVE CONTROL OF THE VISTA F-16	**5. FUNDING NUMBERS**
6. AUTHOR(S) Michael J. Stepaniak, Second Lieutenant, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/95D-26
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Captain Peter K. Eide, WL/FIGS, 2210 Eighth St STE 11, Wright-Patterson AFB, OH 45433-7521	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; Distribution Unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

Multiple model adaptive control (MMAC) is investigated using the high-fidelity, nonlinear, six-degree-of-freedom Simulation Rapid-Prototyping Facility VISTA F-16. Detection of single actuator and sensor failures is considered, with an MMAC algorithm initially pursued which allows a controller specifically designed for each particular failure condition to replace the standard F-16 Block 40 flight control system (FCS) once the failure is detected. The synthesis of certain discrete-time LQG/PI controllers (those using control variables linearly dependent on state derivatives) is shown to be unattainable due to numerical difficulties. A novel control technique, termed control redistribution, is introduced which redistributes control commands (that would normally be sent to failed actuators) to the non-failed actuators, accomplishing the same control action on the aircraft. Multiple model adaptive estimation-based control redistribution is demonstrated to detect single failures in less than one second and to provide a response nearly identical to that anticipated from a fully functional aircraft in the same environment. Moreover, this method directly employs the proven Block 40 FCS, and no other, thereby guaranteeing desirable closed loop performance. A description of modifications necessary for in-flight testing is also provided. This research represents the most realistic simulation of multiple model adaptive control for flight control to date.

14. SUBJECT TERMS Multiple Model Adaptive Control, MMAC, Kalman Filter, LQG, PI, F-16, Control Redistribution, Flight Control, Failure Detection, Reconfigurable Control	**15. NUMBER OF PAGES** 175
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL